

Embedded Android Workshop

ELCE 2014

Karim Yaghmour
@karimyaghmour





These slides are made available to you under a Creative Commons Share-Alike 3.0 license. The full terms of this license are here: <https://creativecommons.org/licenses/by-sa/3.0/>

Attribution requirements and misc., PLEASE READ:

- This slide must remain as-is in this specific location (slide #2), everything else you are free to change; including the logo :-)
- Use of figures in other documents must feature the below “Originals at” URL immediately under that figure and the below copyright notice where appropriate.
- You are free to fill in the “Delivered and/or customized by” space on the right as you see fit.
- You are FORBIDDEN from using the default “About” slide as-is or any of its contents.

(C) Copyright 2010-2014, Opersys inc.

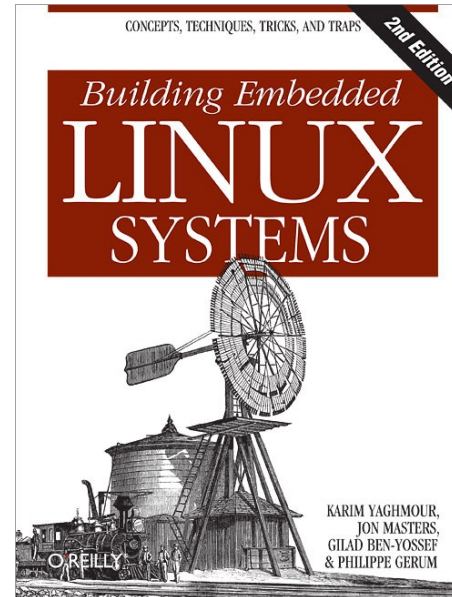
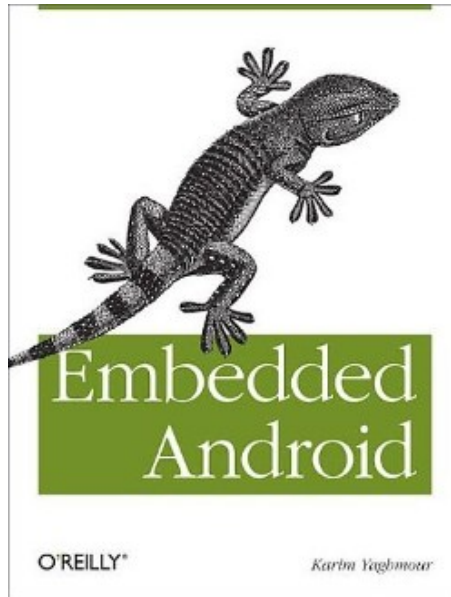
These slides created by: Karim Yaghmour

Originals at: www.opersys.com/community/docs

Delivered and/or customized by

About

- Author of:



- Introduced Linux Trace Toolkit in 1999
- Originated Adeos and relayfs (kernel/relay.c)
- Training, Custom Dev, Consulting, ...

About Android

- Huge
- Fast moving
- Stealthy

Introduction to Embedded Android

- Basics
- History
- Ecosystem
- Legal framework
- Platform and hardware requirements
- Development tools

1. Basics

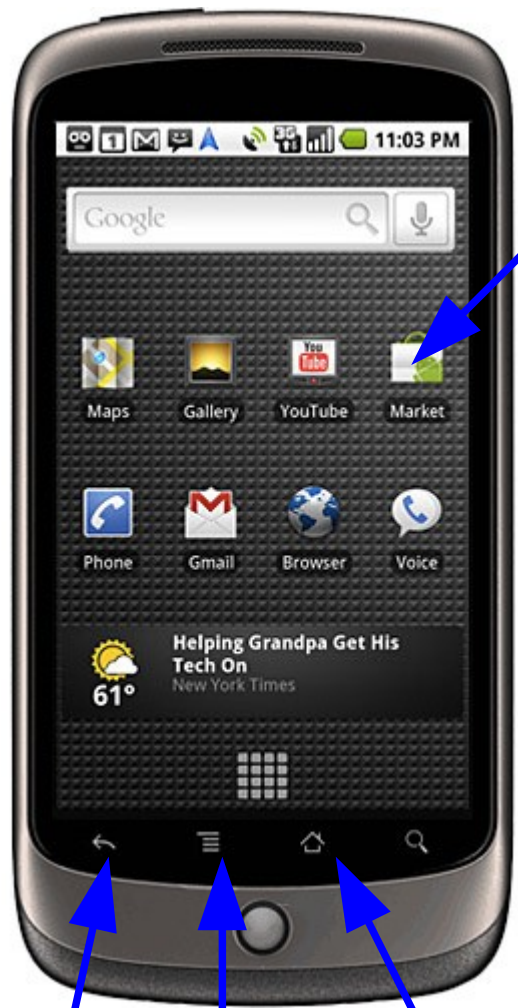
- Features
- UX Concepts
- App Concepts

1.1. Features (old snapshot)

- Application framework enabling reuse and replacement of components
- Dalvik virtual machine optimized for mobile devices
- Integrated browser based on the open source WebKit engine
- Optimized graphics powered by a custom 2D graphics library; 3D graphics based on the OpenGL ES 1.0 specification (hardware acceleration optional)
- SQLite for structured data storage
- Media support for common audio, video, and still image formats (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
- GSM Telephony (hardware dependent)
- Bluetooth, EDGE, 3G, and WiFi (hardware dependent)
- Camera, GPS, compass, and accelerometer (hardware dependent)
- Rich development environment including a device emulator, tools for debugging, memory and performance profiling, and a plugin for the Eclipse IDE

1.2. UX Concepts

- Browser-like
- iPhone-ish
- No user-concept of “task”
- Main keys:
 - HOME
 - SEARCH
 - BACK
 - MENU
- App-model allows users to safely install/test almost anything



Activity

“Click”

Activity #1

“Back”

“Click”

Activity #2

“Click”

“Back”

Activity #3

“Home”

Home

Menu

Back

1.3. App Concepts

- No single entry point (No main() !?!?)
- Unlike Windows or Unix API/semantics in many ways
- Processes and apps will be killed at random: developer must code accordingly
- UI disintermediated from app “brains”
- Apps are isolated, very
- Behavior predicated on low-memory conditions

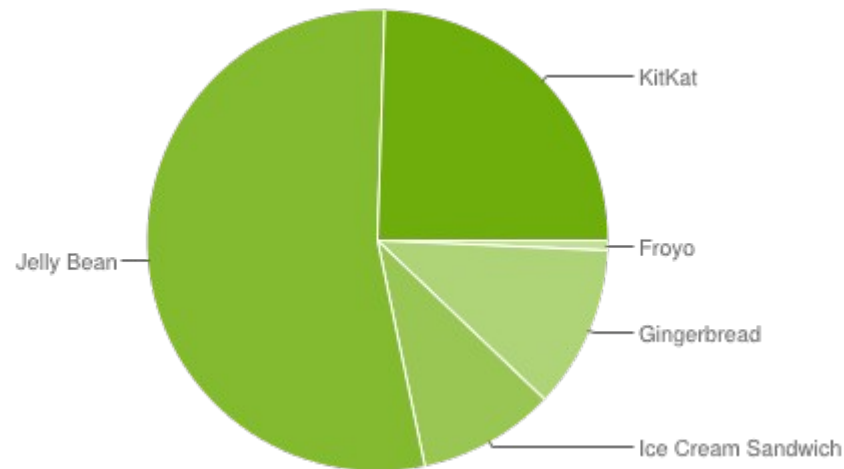
2. History

- 2002:
 - Sergey Brin and Larry Page started using Sidekick smartphone
 - Sidekick one of 1st smartphones integrating web, IM, mail, etc.
 - Sidekick was made by Danger inc., co-founded by Andy Rubin (CEO)
 - Brin/Page met Rubin at Stanford talk he gave on Sidekick's development
 - Google was default search engine on Sidekick
- 2004:
 - Despite cult following, Sidekick wasn't making \$
 - Danger inc. board decided to replace Rubin
 - Rubin left. Got seed \$. Started Android inc. Started looking for VCs.
 - Goal: Open mobile hand-set platform
- 2005 - July:
 - Got bought by Google for undisclosed sum :)
- 2007 - November:
 - Open Handset Alliance announced along with Android

- 2008 - Sept: Android 1.0 is released
- 2009 - Feb.: Android 1.1
- 2009 - Apr: Android 1.5 / Cupcake
- 2009 - Sept.: Android 1.6 / Donut
- 2009 - Oct: Android 2.0/2.1 / Eclair
- 2010 - May: Android 2.2 / Froyo
- 2010 - Dec: Android 2.3 / Gingerbread
- 2011 - Jan : Android 3.0 / Honeycomb – Tablet-optimized
- 2011 – May: Android 3.1 – USB host support
- 2011 – Nov: Android 4.0 / Ice-Cream Sandwich – merge Gingerbread and Honeycomb
- 2012 – Jun: Android 4.1 / Jelly Bean – Platform Optimization
- 2012 – Nov: Android 4.2 / Jelly Bean - Multi-user support
- 2013 -- July: Android 4.3 / Jelly Bean - GL ES 3.0
- 2013 -- Oct: Android 4.4 / Kit Kat - Low-memory support
- 2014 -- ??? : Android ?? / L* - ART and other pixie dust

3. Ecosystem

- More than a Billion devices activated
- 1.3M apps (vs. 1.2M for Apple's app store)
- 85% global smartphone marketshare (iOS is 12%)



3.1. Who's playing?

- Leading IP:
 - Google
- Semiconductor manufacturers:
 - ARM, Intel, Freescale, TI, Qualcomm, NVIDIA, ...
- Handset manufacturers:
 - Motorola, Samsung, HTC, LG, Sony Ericsson, ...
- Tablet manufacturers:
 - Motorola, Samsung, Archos, DELL, ASUS, ...
- Special-purpose devices:
 - Nook, Joint Battle Command-Platform, ...
- App stores:
 - Android Market, Amazon App Store, V CAST Apps, B&N NOOK Apps, ...

3.2. Open Handset Alliance

- “... a group of 80 technology and mobile companies who have come together to accelerate innovation in mobile and offer consumers a richer, less expensive, and better mobile experience. Together we have developed Android™, the first complete, open, and free mobile platform.”
- Unclear what OHA does or what benefits, if any, members derive
- Not an organization with board members, staff, etc. ... just an “Alliance”
- Google's Android team are the lead on all bleeding edge dev, all else tag along
- OHA is largely inactive / absent
- Comprised of:
 - Mobile Operators: Sprint, T-Mobile, Vodafone, NTT Docomo, ...
 - Handset Manufacturers: HTC, Motorola, LG, Samsung, Sony Ericsson, ...
 - Semiconductor Companies: ARM, Freescale, Intel, NVIDIA, Qualcomm, TI, ...
 - Software Companies: Google, ...
 - Commercialization Companies: ...

4. Legal Framework

- Code access
- Code licenses
- Branding use
- Google's own Android Apps
- Alternative App stores
- Oracle v. Google

4.1. Code Access

- Parts:
 - Kernel
 - Android Open Source Project (AOSP)
- Kernel:
 - Should have access to latest shipped version => GPL requirement
 - Google-maintained forks at android.googlesource.com
- AOSP:
 - *Usually* Code-drops every 6 months
 - Official AOSP branches at android.googlesource.com
 - Managed by “repo” tool, an overlay to “git”
- Honeycomb (3.0) code required talking to Google
 - ... which hasn't precluded moders from lifting binaries off the SDK and putting Honeycomb on all sorts of devices, including B&N's Nook ...

4.2. Code Licenses

- Kernel:
 - GNU General Public License (a.k.a. GPL)
- AOSP:
 - Mostly Apache License 2.0 (a.k.a. ASL)
 - Having GPL-free user-space was a design goal
 - A few GPL and LGPL parts: mainly BlueZ and DBUS
 - Some key components in BSD: Bionic and Toolbox
 - “external/” directory contains a mixed bag of licenses, incl. lots of GPL
- May be desirable to add GPL/LGPL components:
 - BusyBox
 - uClibc / eglibc / glibc

4.3. Branding Use

- Android Robot:
 - Very much like the Linux penguin
- Android Logo (A-N-D-R-O-I-D w/ typeface):
 - Cannot be used
- Android Custom Typeface:
 - Cannot be used
- Android in Official Names:
 - As descriptor only: “for Android”
 - Most other uses require approval
- Android in Messaging:
 - Allowed if followed by a generic: “Android Application”
- Compliance through CDD/CTS involved in “approval”



4.4. Google's own Android Apps

- The non-AOSP apps:
 - Android Market
 - YouTube
 - Maps and Navigation
 - Gmail
 - Voice
 - SkyMap
 - ...
- Require:
 - CTS/CDD Compliance
 - Signed agreement w/ Google
- Inquiries: android-partnerships@google.com

4.5. Alternative “App Stores”

- A couple of stores are already public:
 - Android Market
 - Amazon App Store
 - V CAST Apps
 - B&N NOOK Apps
 - ...
- Nothing precluding you from having your own

4.6. Oracle v. Google

- Filed August 2010
- Patent infringement:
 - 6,125,447; 6,192,476; 5,966,702; 7,426,720; RE38,104; 6,910,205; and 6,061,520
- Copyright infringement:
- Android does not use any Oracle Java libraries or JVM in the final product.
- Android relies on Apache Harmony and Dalvik instead.
- In October 2010, IBM left Apache Harmony to join work on Oracle's OpenJDK, leaving the project practically orphaned.
- ...
- In Spring of 2012 Oracle lost both on Copyright and Patent fronts

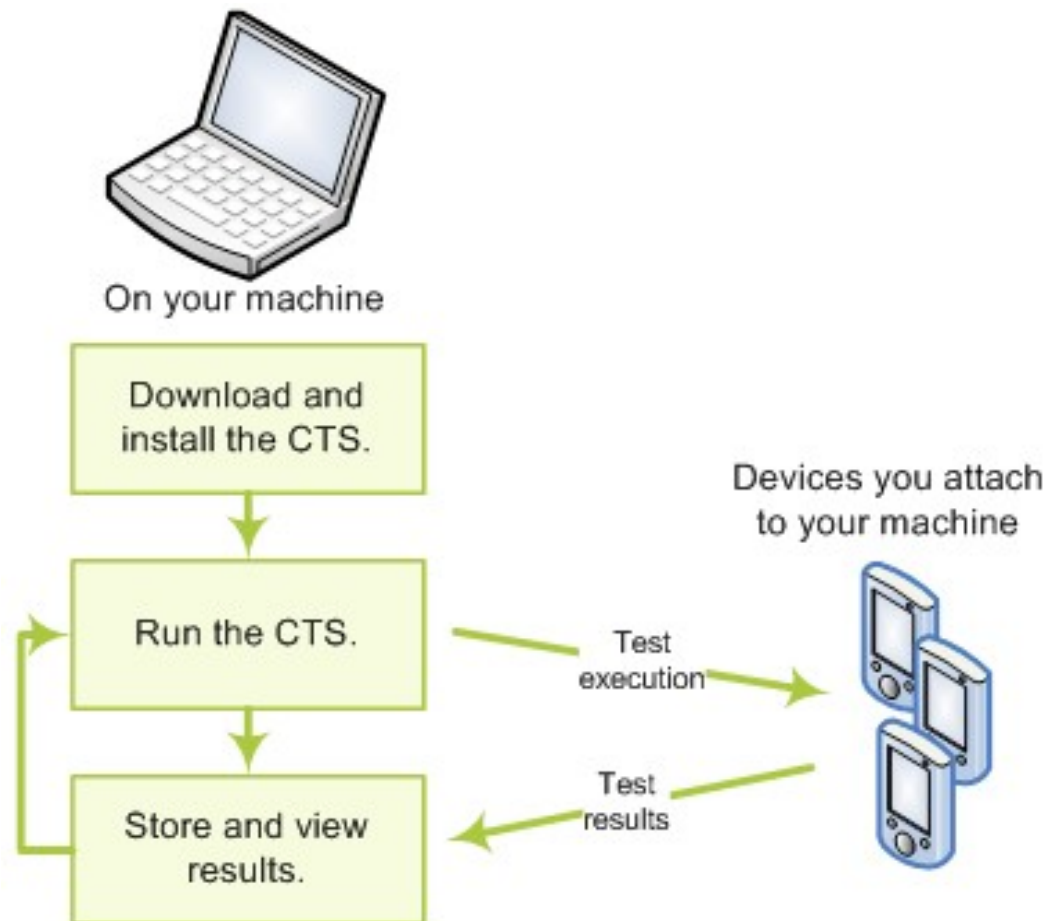
5. Platform and Hardware requirements

- In principle:
 - Android runs on top of Linux
 - Therefore: if it runs Linux, it can run Android
- Known to have been made to work on:
 - ARM
 - x86
 - MIPS
 - SuperH
- Put in all sort of devices:
 - Washers, micro-wave ovens, car systems, etc.

5.1. Compliance Definition Document

- Software: MUST conform to AOSP
- Application Packaging Compatibility: support “.apk” files
- Multimedia Compatibility: decoders, encoders, recording, ...
- Developer Tool Compatibility: adb, ddms, Monkey
- Hardware compatibility:
 - Display and Graphics
 - Input Devices
 - Data Connectivity
 - Cameras
 - Memory and Storage
 - USB
- Performance Compatibility
- Security Model Compatibility
- Software Compatibility Testing
- Updatable Software: MUST include mechanism to update

5.2. Compatibility Test Suite



6. Development tools

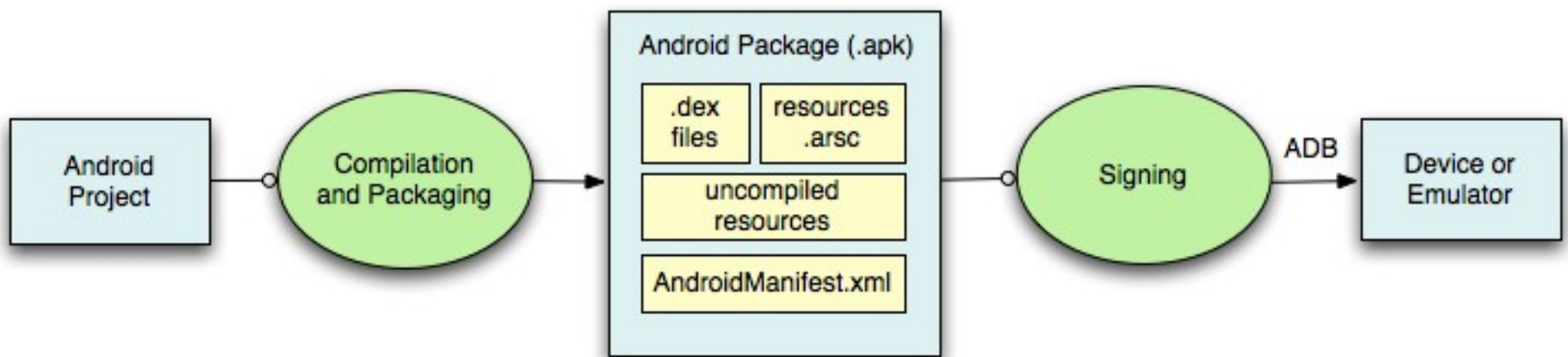
- Requirements
- App dev tools and resources
- App debugging

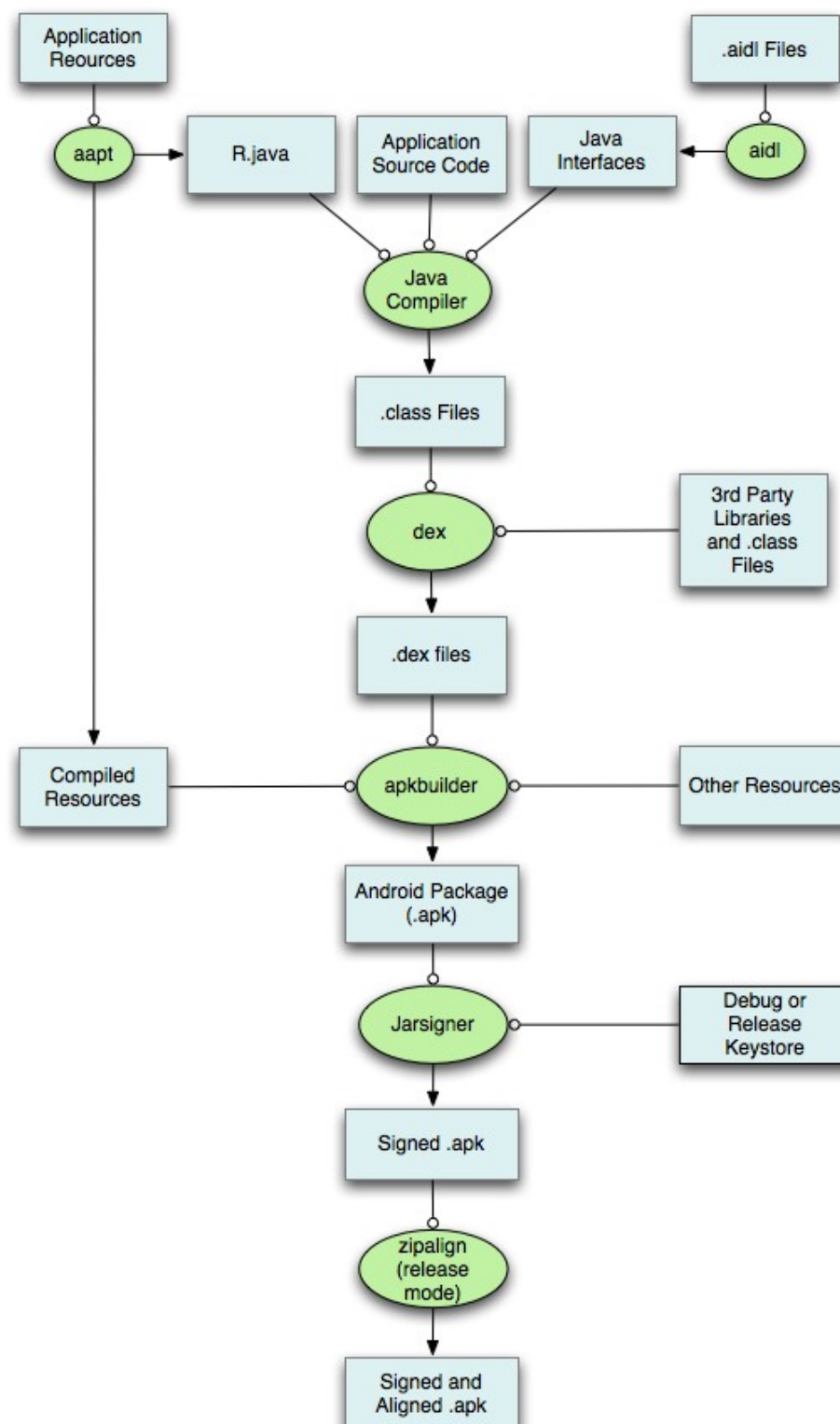
6.1. Requirements

- App development and debugging:
 - Windows / Mac / Linux workstation
 - JDK
 - Eclipse w/ ADT plugin
 - Highly recommended: real device(**S**)
- Platform development:
 - GNU cross-dev toolchain
 - JTAG debugger
 - ... more on this later

6.2. App dev tools and resources

- SDK:
 - android – manage AVDs and SDK components
 - apkbuilder – creating .apk packages
 - dx – converting .jar to .dex
 - adb – debug bridge
 - ...
- Emulator – QEMU-based ARM emulator
 - Use KVM for x86 instead
- NDK: GNU toolchain for native binaries
- Documentation: developer.android.com





6.3. App debugging

- adb
- ddms
- monkeyrunner
- traceview
- logcat
- Eclipse integration (ADT plugin)

Concepts and Internals

1. Android Concepts
2. Framework Intro
3. Overall Architecture
4. System startup
5. Linux Kernel
6. Hardware Support
7. Native User-Space
8. Dalvik
9. JNI
10. System Server
11. Calling on Services
12. Activity Manager
13. Binder
14. HAL
15. Stock AOSP Apps

1. Android Concepts

- Components
- Intents
- Component lifecycle
- Manifest file
- Processes and threads
- Remote procedure calls

1.1. Components

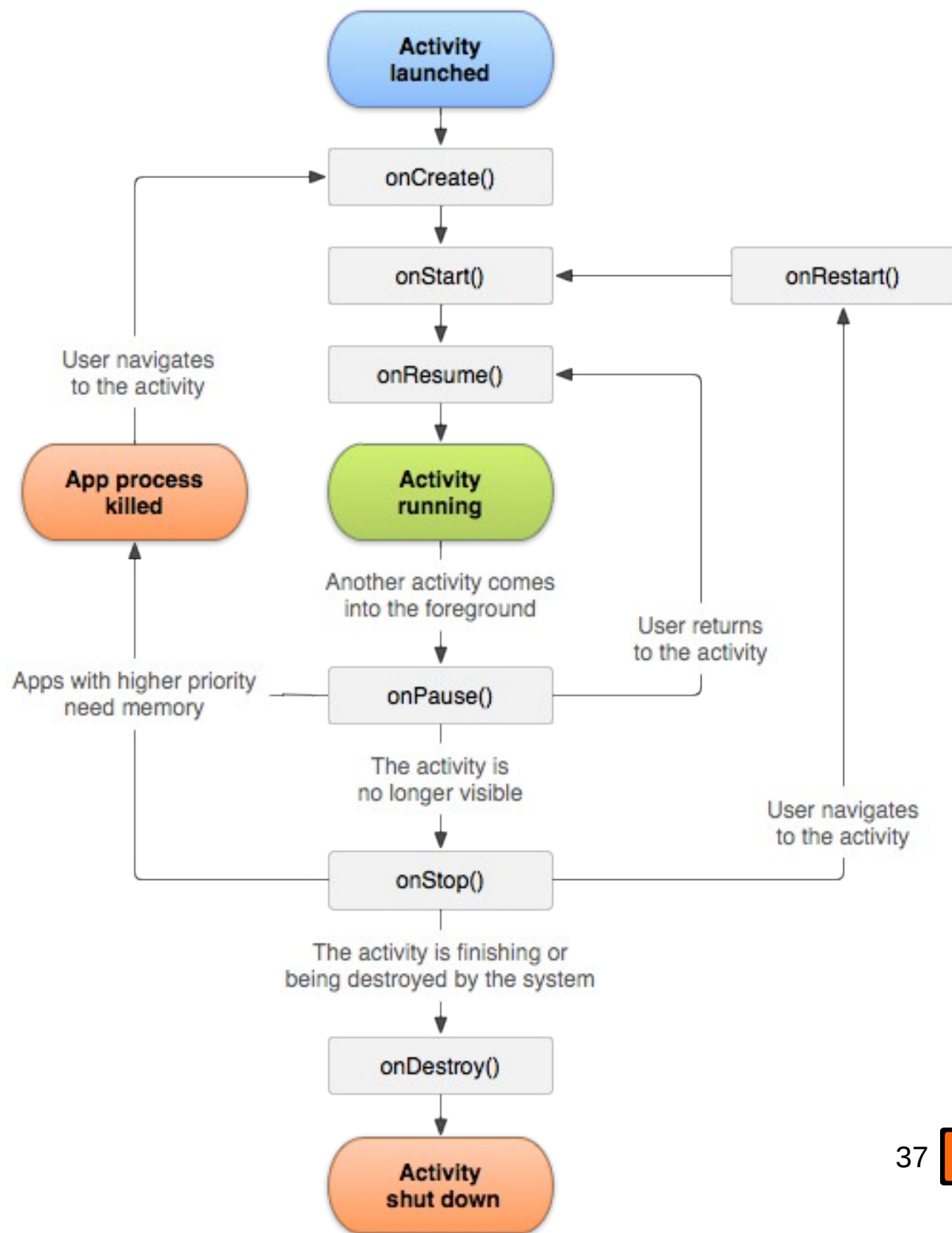
- 1 App = N Components
- Apps can use components of other applications
- App processes are automatically started whenever any part is needed
- Ergo: N entry points, !1, and !main()
- Components:
 - Activities
 - Services
 - Broadcast Receivers
 - Content Providers

1.2. Intents

- Intent = asynchronous message w/ or w/o designated target
- Like a polymorphic Unix signal, but w/o required target
- Intents “payload” held in Intent Object
- Intent Filters specified in Manifest file

1.3. Component lifecycle

- System automatically starts/stops/kills processes:
 - Entire system behaviour predicated on low memory
- System triggers Lifecycle callbacks when relevant
- Ergo: Must manage Component Lifecycle
- Some Components are more complex to manage than others



1.4. Manifest file

- Informs system about app's components
- XML format
- Always called AndroidManifest.xml
- Activity = `<activity> ... static`
- Service = `<service> ... static`
- Broadcast Receiver:
 - Static = `<receiver>`
 - Dynamic = `Context.registerReceiver()`
- Content Provider = `<provider> ... static`

1.5. Processes and threads

- Processes
 - Default: all callbacks to any app Component are issued to the main process thread
 - <activity>—<service>—<recipient>—<provider> have process attribute to override default
 - Do NOT perform blocking/long operations in main process thread:
 - Spawn threads instead
 - Process termination/restart is at system's discretion
 - Therefore:
 - Must manage Component Lifecycle
- Threads:
 - Create using the regular Java Thread Object
 - Android API provides thread helper classes:
 - Looper: for running a message loop with a thread
 - Handler: for processing messages
 - HandlerThread: for setting up a thread with a message loop

1.6. Remote procedure calls

- Android RPCs = Binder mechanism
- No Sys V IPC due to in-kernel resource leakage
- Binder is a low-level functionality, not used as-is
- Instead: must define interface using Interface Definition Language (IDL)
- IDL fed to aidl Tool to generate Java interface definitions

2. Framework Introduction

- UI
- Data storage
- Security/Permissions
- ... and much more ... :
 - Graphics
 - Audio and Video
 - Location and Maps
 - Bluetooth
 - NFC

2.1. UI

- Everything based on hierarchy of Views and ViewGroups (layouts)
- Declared in XML or dynamically through Java
- UI components:
 - Widgets
 - Event handlers
 - Menus
 - Dialogs
 - Notifications
 - ...

2.2. Data storage

- Shared preferences
 - Private primitive key-pair values
- Internal storage
 - Private data on device memory
- External storage
 - Public data on shared external device (SD)
- SQLite DB
 - Private DB
- Network connection
 - Web-based storage (REST)

2.3. Security/Permissions

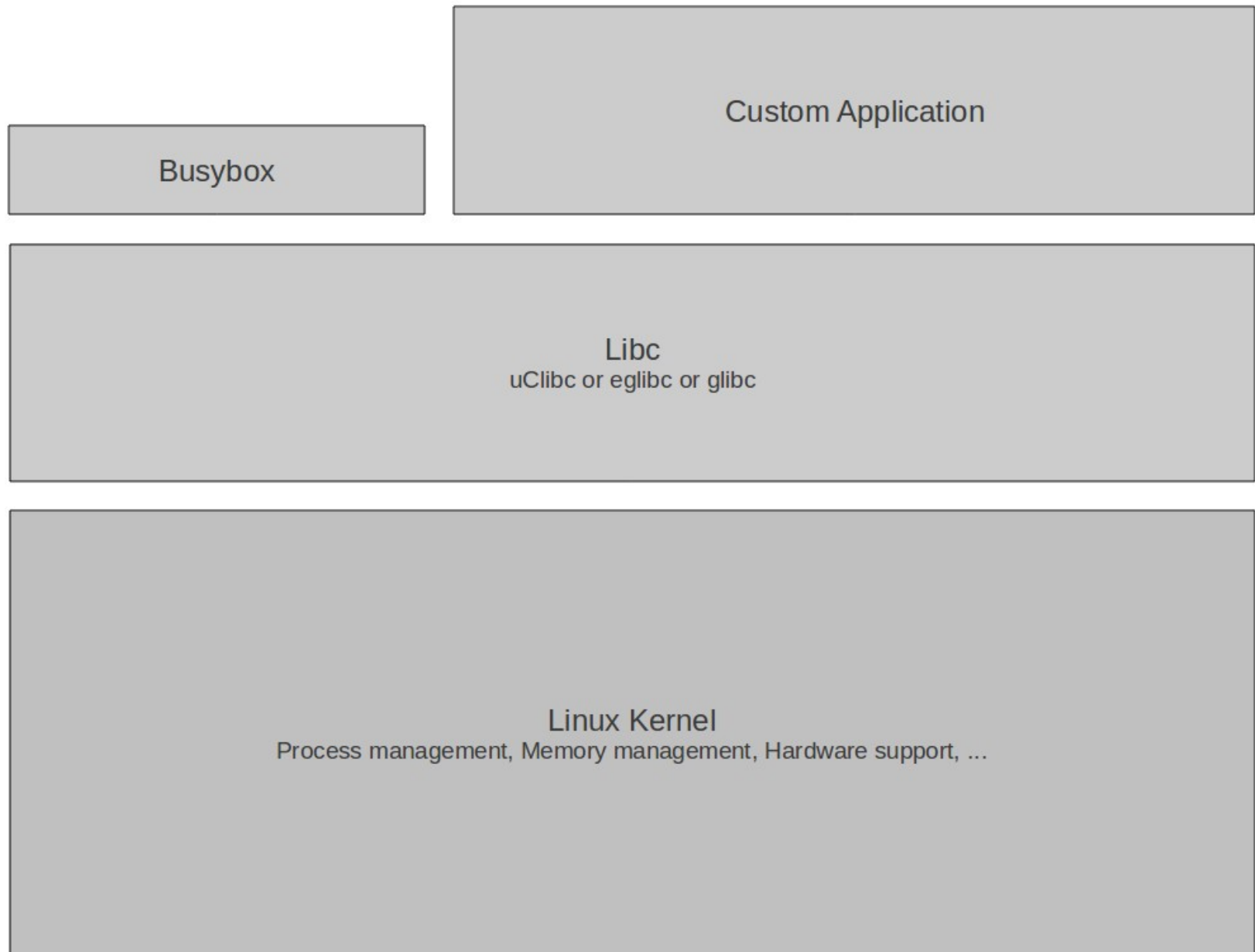
- Most security enforced at process level: UID, GID
- Permissions enforce restrictions on:
 - Per-process operations
 - Per-URI access
- Applications are sandboxed
- Specific permissions required to “exit” sandbox
- Decision to grant access based on:
 - Certificates
 - User prompts
- All permissions must be declared statically

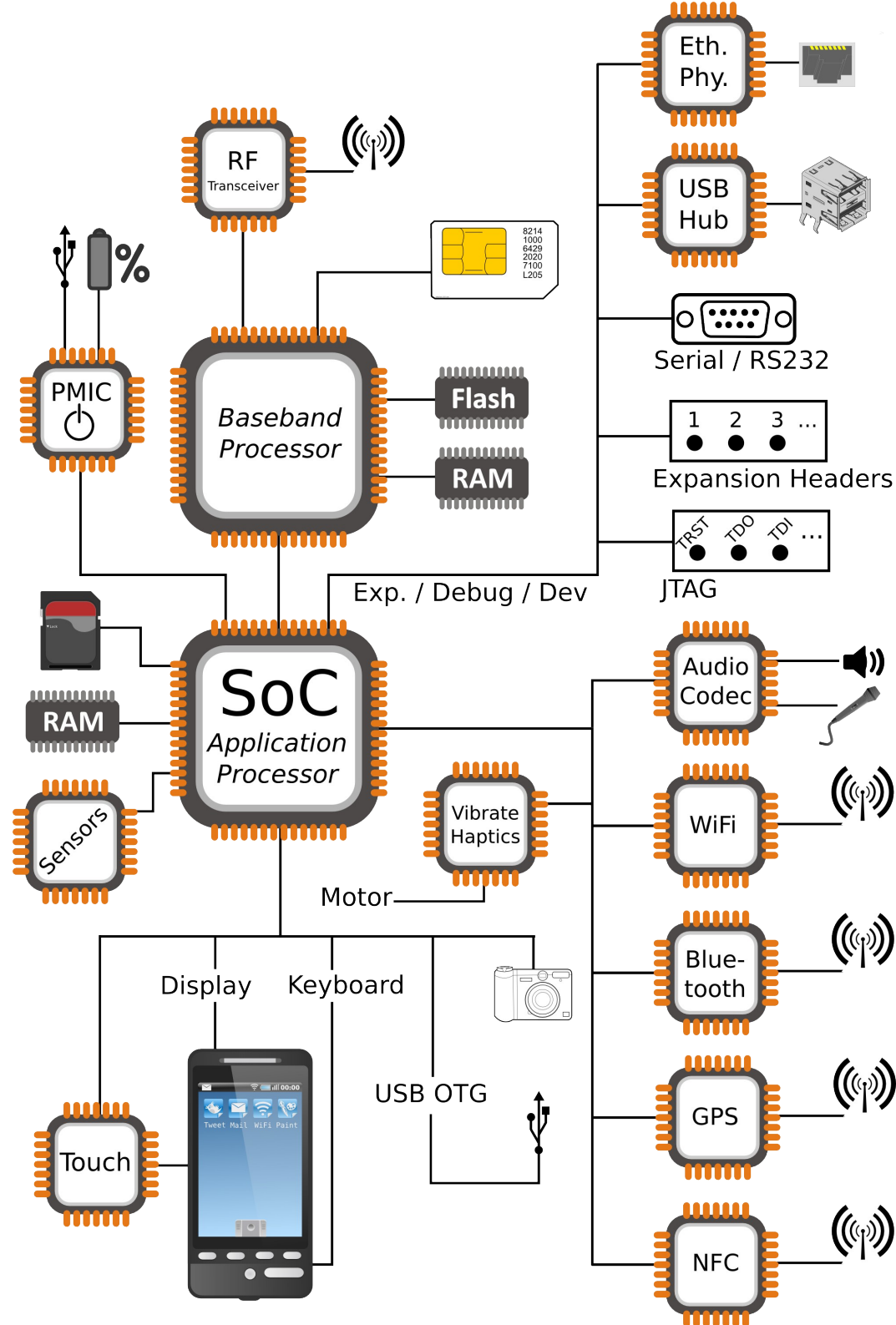
2.4. Native development

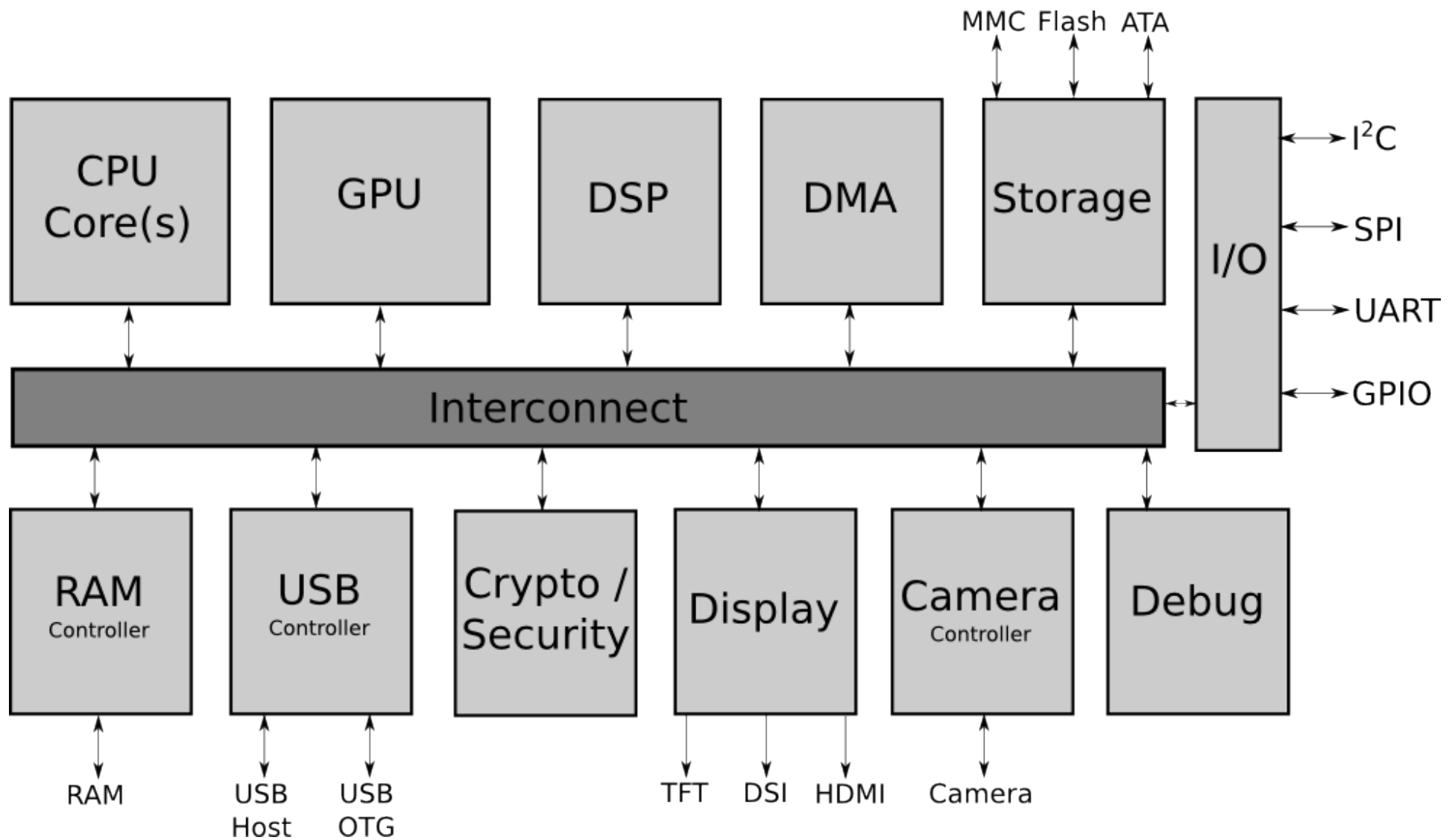
- Useful for:
 - Porting existing body of code to Android
 - Developing optimized native apps, especially for gaming
- Provides:
 - Tools and build files to generate native code libraries from C/C++
 - Way to embed native libs into .apk
 - Set of stable (forward-compatible) native libs
 - Documentation, samples and tutorials
- Enables:
 - Calling native code from Java using JNI
 - Implementing fully native apps (since 2.3)
- Doesn't allow you to:
 - Compile traditional Linux/Unix apps as-is

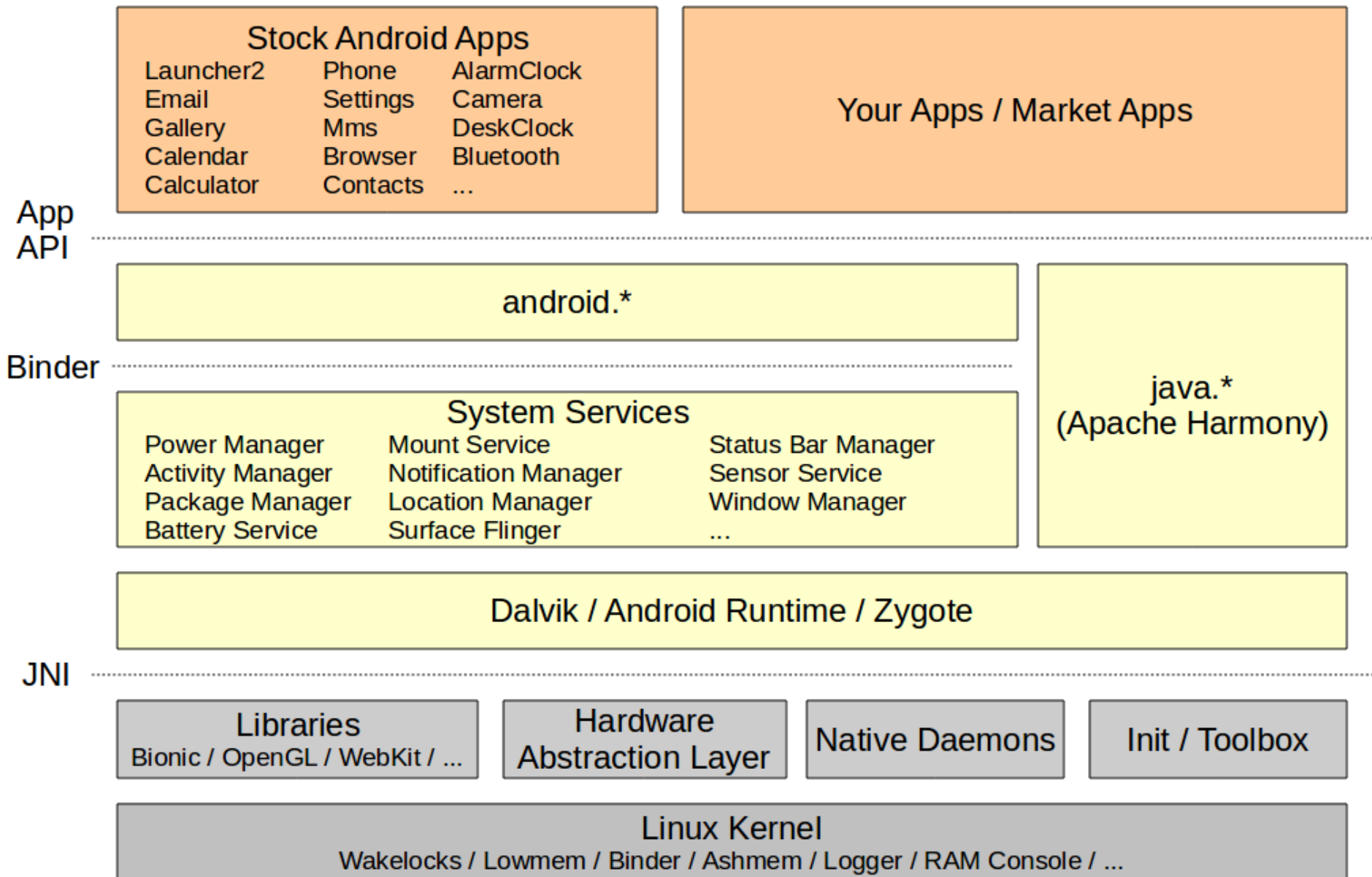
3. Architecture

- Embedded Linux
- Modern phone/tablet
- System-on-Chip (SoC)
- Android



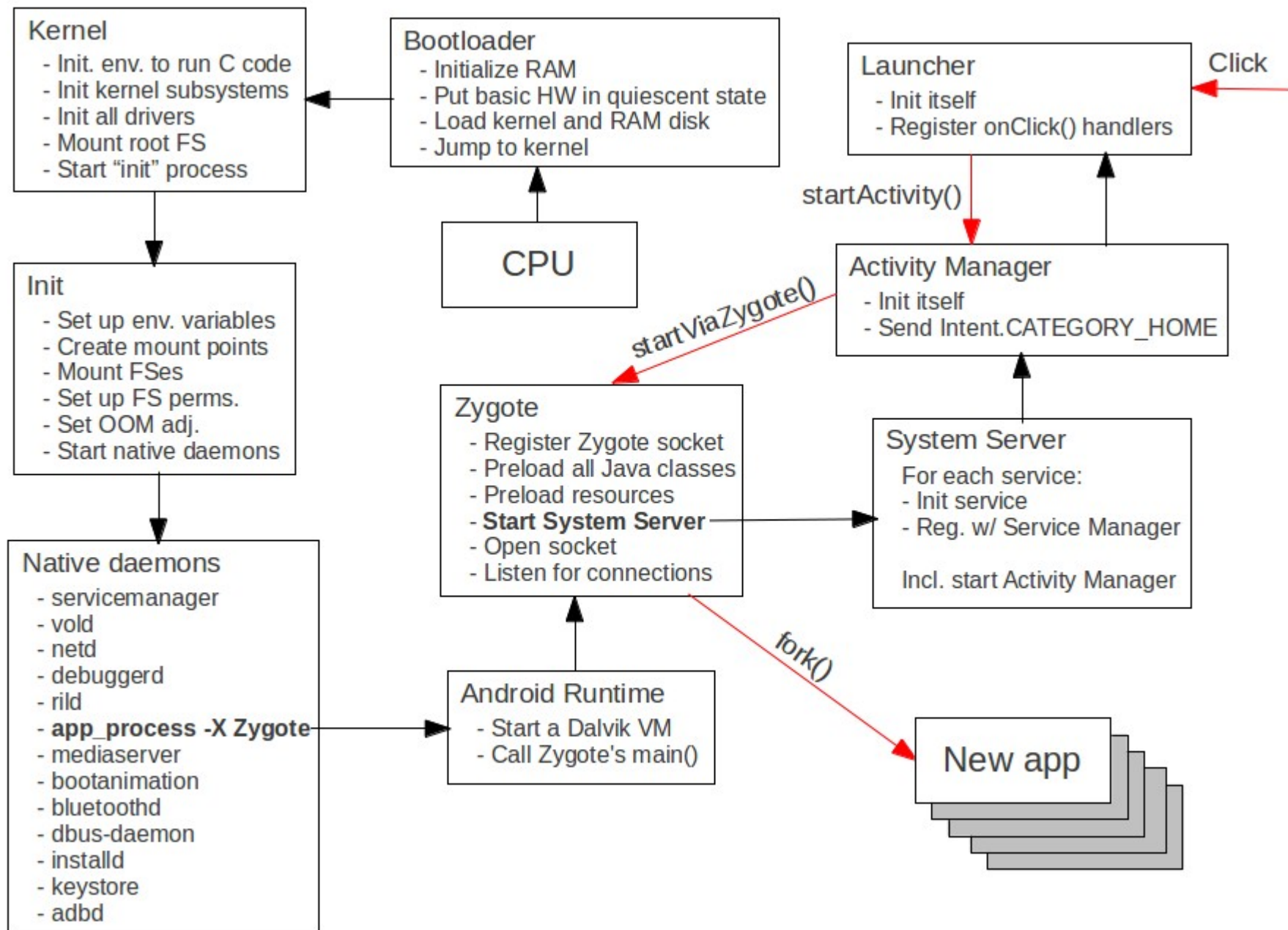






4. System Startup

- Bootloader
- Kernel
- Init
- Zygote
- System Server
- Activity Manager
- Launcher (Home)



4.1. Bootloader

- aosp/bootable/bootloader
 - Custom bootloader for Android
 - USB-based
 - Implements the “fastboot” protocol
 - Controlled via “fastboot” cli tool on host
- aosp/bootable/recovery
 - UI-based recovery boot program
 - Accessed through magic key sequence at boot
 - Usually manufacturer specific variant

- Flash layout:

0x000003860000-0x000003900000	:	"misc"		
0x000003900000-0x000003e00000	:	"recovery"		
0x000003e00000-0x000004300000	:	"boot"	←	Kernel
0x000004300000-0x00000c300000	:	"system"	←	/system
0x00000c300000-0x0000183c0000	:	"userdata"	←	/data
0x0000183c0000-0x00001dd20000	:	"cache"	←	/cache
0x00001dd20000-0x00001df20000	:	"kpanic"		
0x00001df20000-0x00001df60000	:	"dinfo"		
0x00001df60000-0x00001dfc0000	:	"setupdata"		
0x00001dfc0000-0x00001e040000	:	"splash1"		
0x000000300000-0x000001680000	:	"modem"		

From Acer Liquid-E

4.2. Kernel

- Early startup code is very hardware dependent
- Initializes environment for the running of C code
- Jumps to the architecture-independent `start_kernel()` function.
- Initializes high-level kernel subsystems
- Mounts root filesystem
- Starts the init process

4.3. Android Init

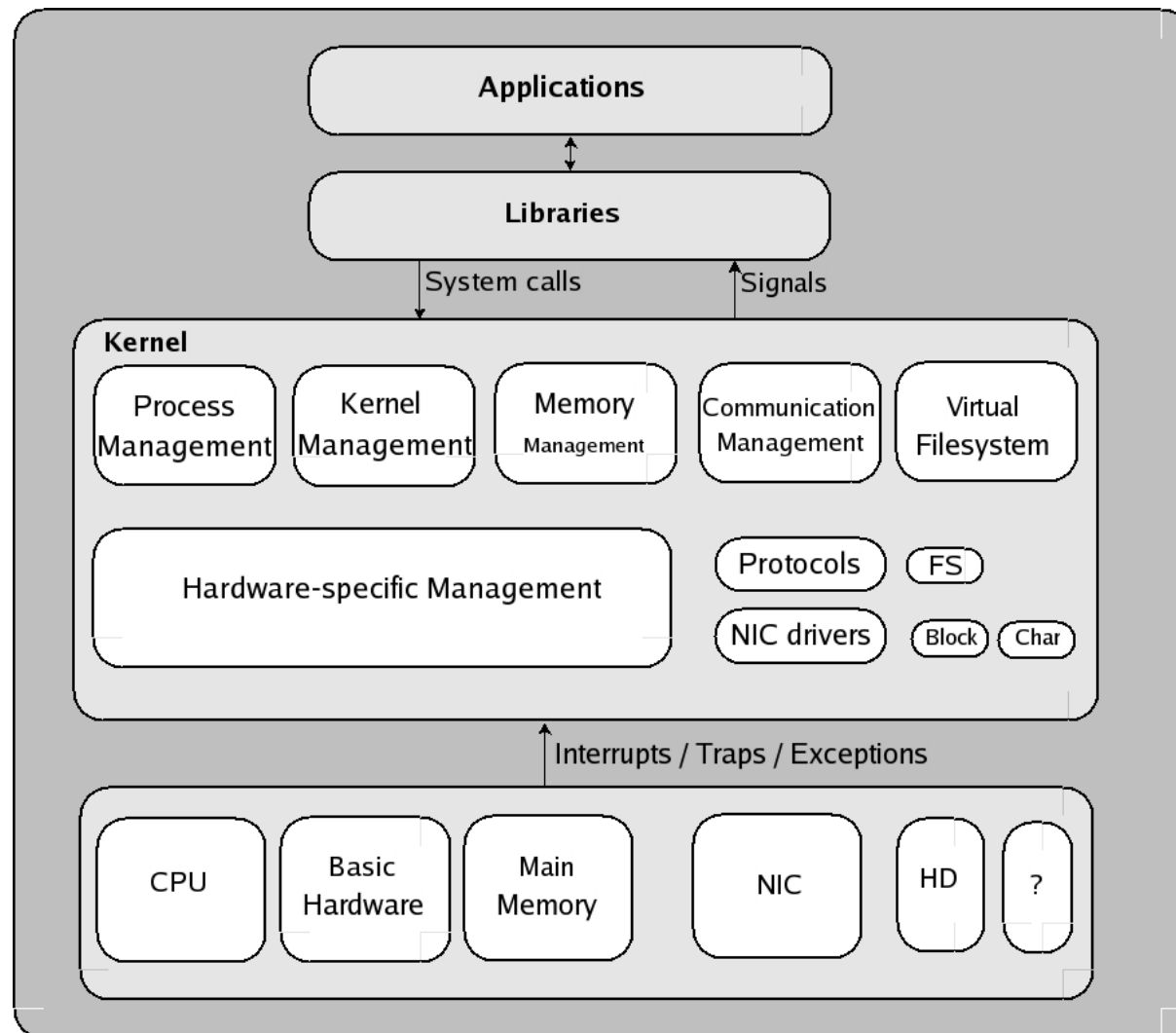
- Open, parses, and runs /init.rc:
 - Create mountpoints and mount filesystems
 - Set up filesystem permissions
 - Set OOM adjustments properties
 - Start daemons:
 - adbd
 - servicemanager (binder context manager)
 - vold
 - netd
 - rild
 - app_process -Xzygote (Zygote)
 - mediaserver
 - ...

4.4. Zygote, etc.

- Init:
 - `app_process -Xzygote (Zygote)`
- `frameworks/base/cmds/app_process/app_main.cpp`:
 - `runtime.start("com.android.internal.os.Zygote", ...`
- `frameworks/base/core/jni/AndroidRuntime.cpp`:
 - `startVM()`
 - Call Zygote's `main()`
- `frameworks/base/core/java/com/android/internal/os/ZygoteInit.java`:
 - ...

- preloadClasses()
- startSystemServer()
- ... magic ...
- Call SystemServer's run()
- frameworks/base/services/java/com/android/server/SystemServer.java:
 - Start **all** system services/managers
 - Start ActivityManager:
 - Send Intent.CATEGORY_HOME
 - Launcher2 kicks in

5. Linux Kernel



5.1. Androidisms

- Wakelocks
- lowmem handler
- Binder
- ashmem – Anonymous Shared Memory
- RAM console
- Logger
- ...

6. Hardware support

Bluetooth	BlueZ through D-BUS IPC (to avoid GPL contamination it seems)
GPS	Manufacturer-provided libgps.so
Wifi	wpa_supplicant
Display	Std framebuffer driver (/dev/fb0)
Keymaps and Keyboards	Std input event (/dev/event0)
Lights	Manufacturer-provided liblights.so
Backlight	
Keyboard	
Buttons	
Battery	
Notifications	
Attention	
Audio	Manufacturer-provided libaudio.so (could use ALSA underneath ... at least as illustrated in their porting guide)
Camera	Manufacturer-provided libcamera.so (could use V4L2 kernel driver underneath ... as illustrated in porting guide)
Power Management	"Wakelocks" kernel patch
Sensors	Manufacturer-provided libsensors.so
Accelerometer	
Magnetic Field	
Orientation	
Gyroscope	
Light	
Pressure	
Temperature	
Proximity	
Radio Layer Interface	Manufacturer-provided libril-<companyname>-<RIL version>.so

7. Native User-Space

- Mainly
 - /data => User data
 - /system => System components
- Also found:
 - /dev
 - /proc
 - /sys
 - /sbin
 - /mnt
 - /cache
 - Etc.

- Libs:

Bionic, SQLite, SSL, OpenGL|ES,

Non-Posix: limited Pthreads support, no SysV IPC

- Toolbox

- Daemons:

servicemanager, vold, rild, netd, adbd, ...

8. Dalvik

- Sun-Java =
Java language + JVM + JDK libs
- Android Java =
Java language + Dalvik + Apache Harmony
- Target:
 - Slow CPU
 - Relatively low RAM
 - OS without swap space
 - Battery powered
- Has had JIT since 2010

8.1. ART

- “Android RunTime”
- 64 bit
- Multi-core
- AOT instead of JIT
- Better debugging capabilities (profiling/stack/crashes)
- Introduced as alternative to Dalvik
- Has now replaced Dalvik

9. JNI – Java Native Interface

- Call gate for other languages, such as C, C++
- Equivalent to .NET's pinvoke
- Usage: include and call native code from App
- Tools = NDK ... samples included
- Check out “*JNI Programmer's Guide and Specification*” - freely available PDF

10. System Services

Entropy Service
Power Manager
Activity Manager
Telephone Registry
Package Manager
Account Manager
Content Manager
System Content Providers
Battery Service
Lights Service
Vibrator Service
Alarm Manager
Init Watchdog
Sensor Service
Window Manager
Bluetooth Service

Device Policy
Status Bar
Clipboard Service
Input Method Service
NetStat Service
NetworkManagement Service
Connectivity Service
Throttle Service
Accessibility Manager
Mount Service
Notification Manager
Device Storage Monitor
Location Manager
Search Service
DropBox Service
Wallpaper Service

Audio Service
Headset Observer
Dock Observer
UI Mode Manager Service
Backup Service
AppWidget Service
Recognition Service
Status Bar Icons
DiskStats Service
ADB Settings Observer

System Services

System Server

Java-built Services

Power Manager	Mount Service
Activity Manager	Notification Manager
Package Manager	Location Manager
Battery Service	Search Service
Window Manager	Wallpaper Service
Status Bar	Headset Observer
Clipboard Service	...

C-built Services

Sensor Service

Surface Flinger

Media Service

Audio Flinger
Media Player Service
Camera Service
Audio Policy Service

Includes:

- StageFright
- Audio effects
- DRM framework

Phone App

JNI

Native Methods for
Java-built Services

Hardware Abstraction Layer

11. Calling on System Services

- Use getSystemService
- Ex: NotificationManager Object reference:

```
String ns = Context.NOTIFICATION_SERVICE;
```

```
NotificationManager mNotificationManager = (NotificationManager) \  
getSystemService(ns);
```

- Prepare your content
- Call on the object:

```
mNotificationManager.notify(HELLO_ID, notification);
```

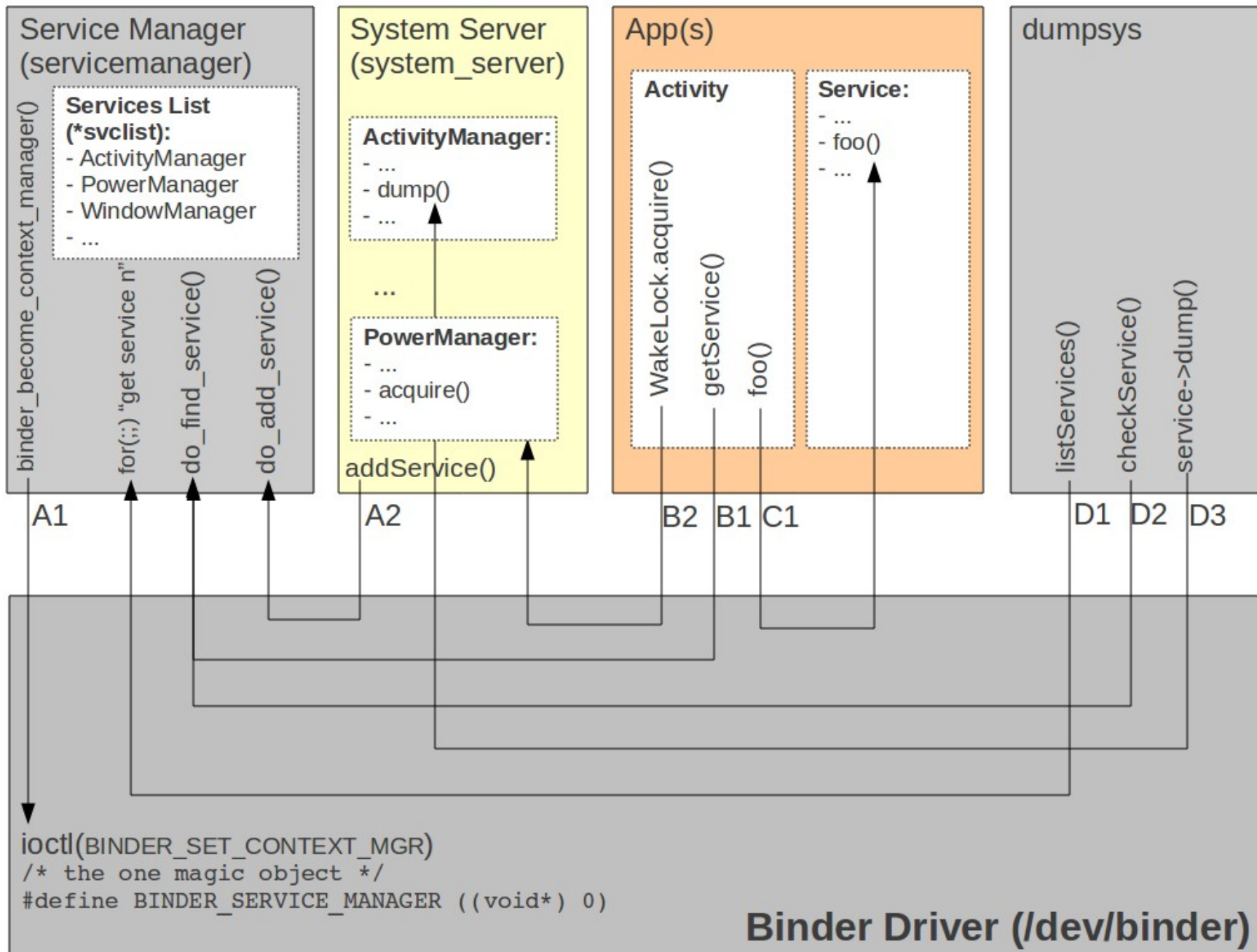
12. ActivityManager

- Start new Activities, Services
- Fetch Content Providers
- Intent broadcasting
- OOM adj. maintenance
- Application Not Responding
- Permissions
- Task management
- Lifecycle management

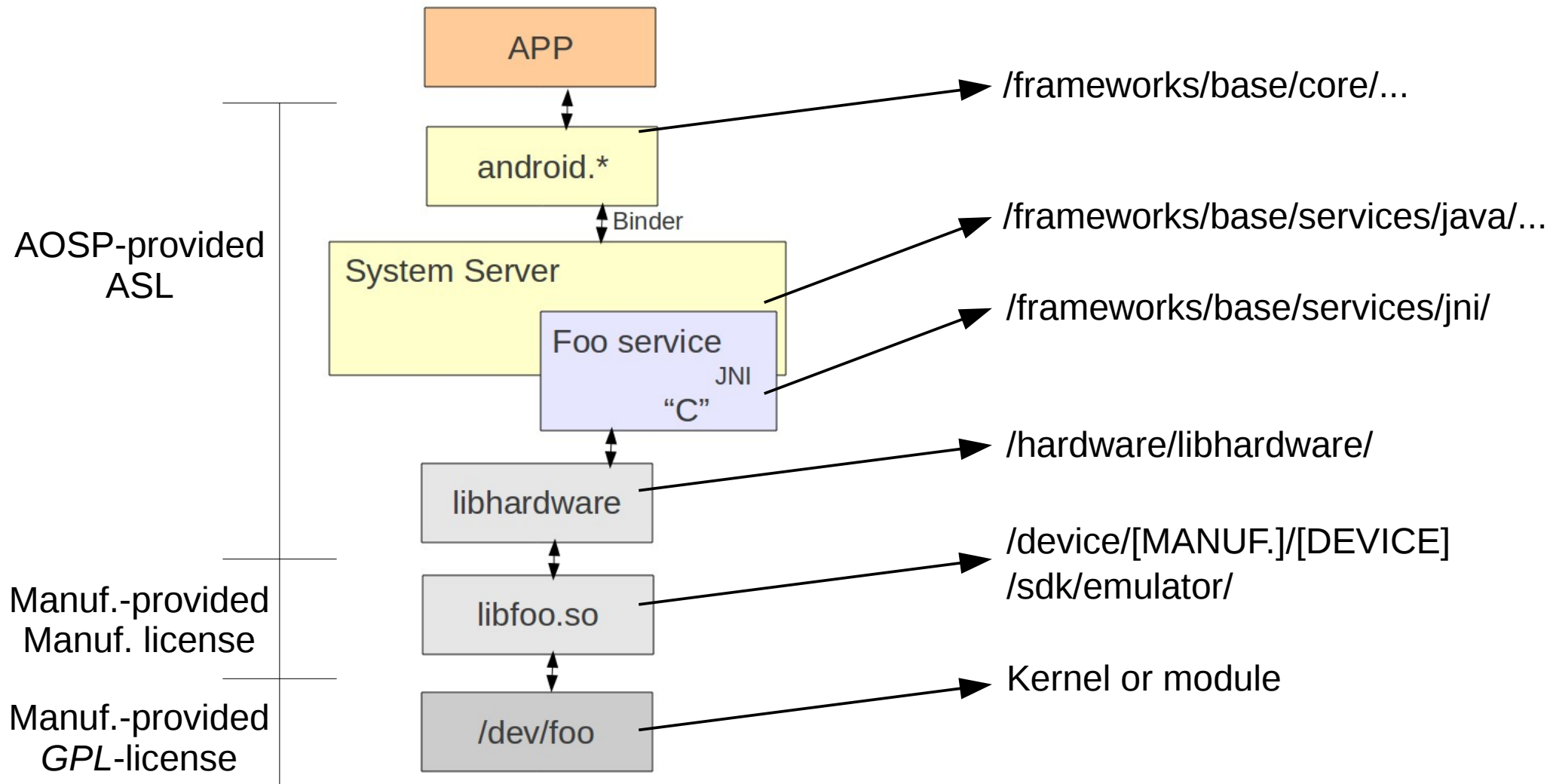
- Ex. starting new app from Launcher:
 - onClick(Launcher)
 - startActivity(Activity.java)
 - *<Binder>*
 - ActivityManagerService
 - startViaZygote(Process.java)
 - *<Socket>*
 - Zygote

13. Binder

- CORBA/COM-like IPC
- Data sent through “parcels” in “transactions”
- Kernel-supported mechanism
- /dev/binder
- Check /proc/binder/*
- android.* API connected to System Server through binder.



14. Hardware Abstraction Layer



15. Stock AOSP Apps

/packages/apps

AccountsAndSettings
AlarmClock
Bluetooth
Browser
Calculator
Calendar
Camera
CertInstaller
Contacts
DeskClock
Email
Gallery
HTMLViewer

Launcher2
Mms
Music
PackageInstaller
Protips
Provision
QuickSearchBox
Settings
SoundRecorder
SpeechRecorder
Stk
VoiceDialer

/packages/providers

ApplicationProvider
CalendarProvider
ContactsProvider
DownloadProvider
DrmProvider
GoogleContactsProvider
MediaProvider
TelephonyProvider
UserDictionaryProvider

/packages/inputmethods

LatinIME
OpenWnn
PinyinIME

Working with the Android Open Source Project

- Tools and location
- Content
- Building
- Build tricks
- Build system architecture
- Output images
- Using adb
- Custom toolchains and dev kits
- Compatibility Test Suite
- Basic hacks

1. Tools and location

- Location:
 - <http://android.googlesource.com/>
- Get “repo”:

```
$ curl https://dl-ssl.google.com/dl/googlesource/git-repo/repo > ~/bin/repo
```

```
$ chmod a+x ~/bin/repo
```
- Fetch the AOSP:
 - Make sure you fetch a tagged release
 - Gingerbread:

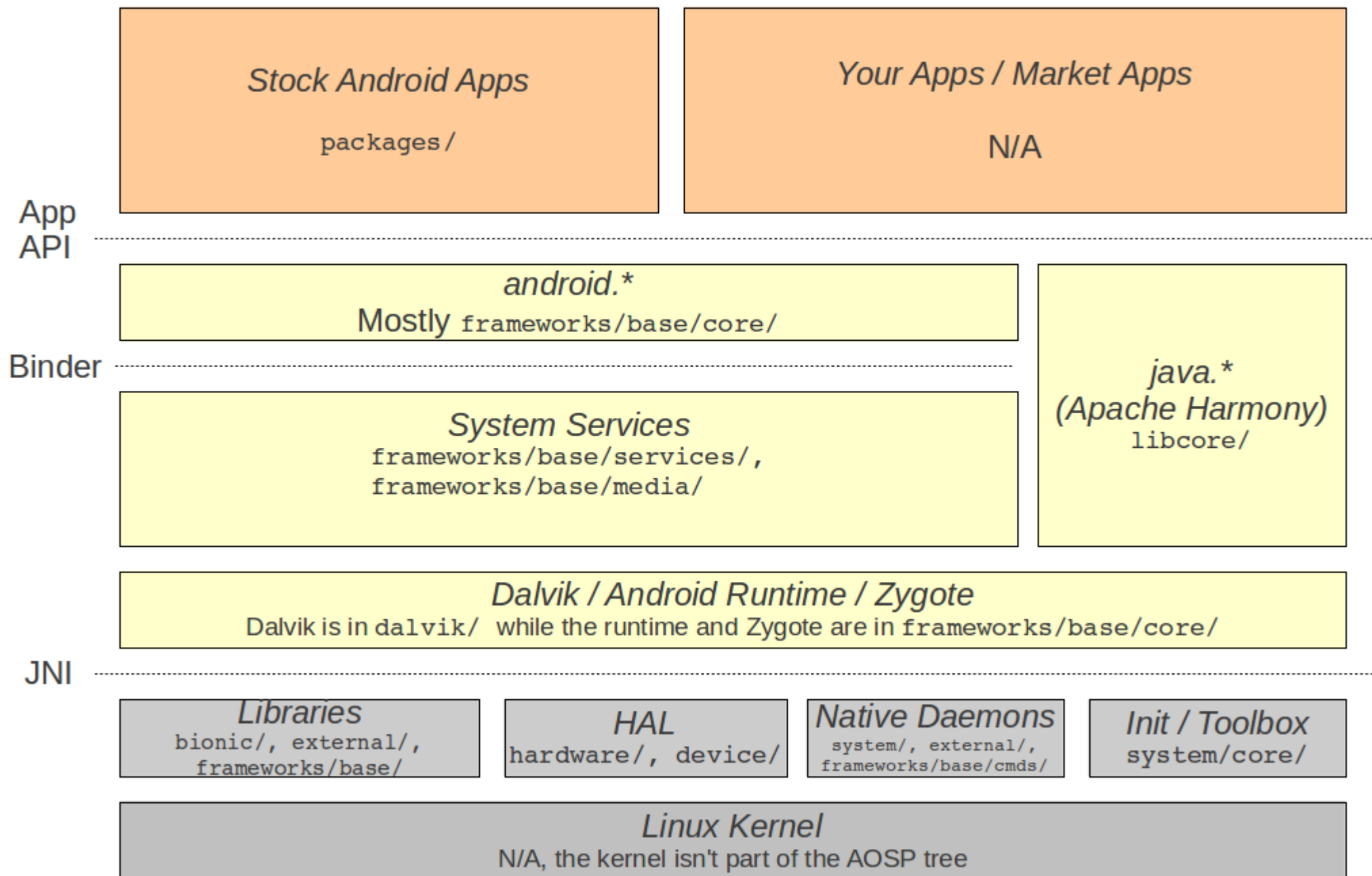
```
$ repo init -u https://android.googlesource.com/platform/manifest
```

```
-b android-2.3.7_r1
```

```
$ repo sync
```

2. Content

bionic	C library replacement
bootable	Reference bootloader
build	Build system
cts	Compatibility Test Suite
dalvik	Dalvik VM
development	Development tools
device	Device-specific files and components
external	Copy of external projects used by AOSP
frameworks	System services, android.*, Android-related cmds, etc.
hardware	Hardware support libs
libcore	Apache Harmony
ndk	The NDK
packages	Stock Android apps, providers, etc.
prebuilt	Prebuilt binaries
sdk	The SDK
system	pieces of the world that are the core of the embedded linux platform at the heart of Android.



3. Building

- Requires 64-bit Ubuntu 10.04

- Packages required:

```
$ sudo apt-get install build-essential libc6-dev \
> ia32-libs lib32z1 bison flex gperf git-core \
> g++ libc6-dev-i386 libz-dev libx11-dev \
> libstdc++6 lib32ncurses5 lib32ncurses5-dev \
> g++-multilib
```

- Possibly fix a few symbolic links:

```
$ sudo ln -s /usr/lib32/libstdc++.so.6 /usr/lib32/libstdc++.so
```

```
$ sudo ln -s /usr/lib32/libz.so.1 /usr/lib32/libz.so
```


- Set up build environment:

```
$ ./build/envsetup.sh
```

```
$ lunch
```

- Launch build and go watch tonight's hockey game:

```
$ make -j2
```

- ... though you should check your screen at breaks ...

- Just launch emulator when it's done:

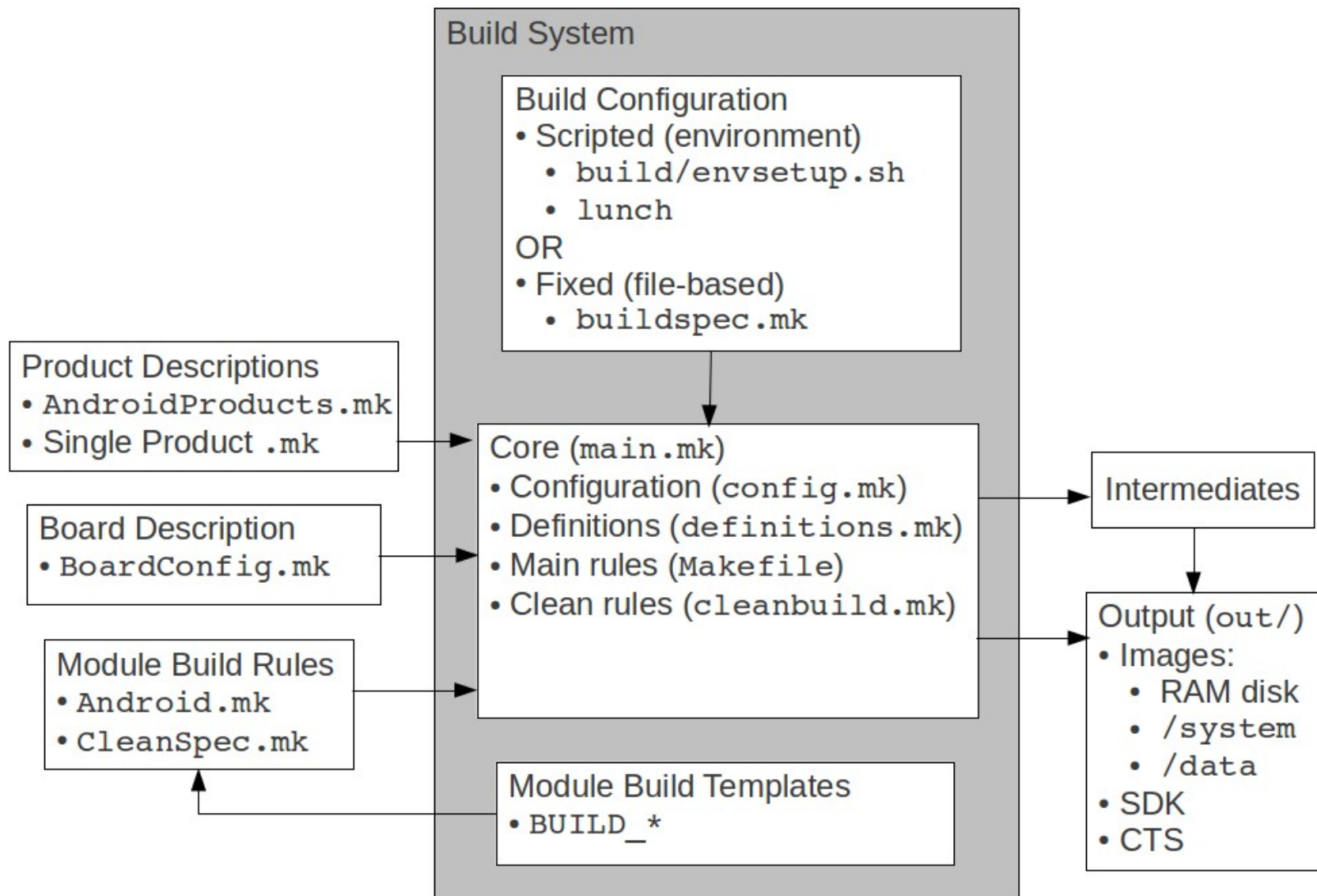
```
$ emulator &
```

- Need to reuse envsetup.sh and lunch on every new shell

4. Build Tricks

- Commands (from build/envsetup.sh):
 - godir
 - croot
 - mm
 - m
- Speeding up the Build:
 - CPU
 - RAM
 - SSD
 - CCACHE
 - `$ export USE_CCACHE=1`

5. Build System Architecture



6. Output Images

- All output and build in [aosp]/out/
- Images at [aosp]/out/target/product/generic/:
 - ramdisk.img
 - system.img
 - userdata-qemu.img
- Kernel is in:
 - prebuilt/android-arm/kernel/kernel-qemu
- Emulator overrides:
 - -kernel
 - -initrd

7. Using adb

- Can use to control/interface w/ running AOSP, including emulator.

- Shell:

\$ adb shell

#

Host

Target

- Dumping the log:

\$ adb logcat

- Copying files to/from target:

\$ adb push foo /data/local

\$ adb pull /proc/config.gz

8. Custom Toolchains and Dev Kits

- Rationale
- SDK generation
- NDK generation

8.1. Rationale

- SDK:
 - Providing other internal teams or external developers access to your modified/custom Android APIs.
- NDK:
 - Same as SDK rationale
- Custom cross-dev toolchain:
 - To avoid having to use a binary toolchain from 3rd party.
 - To control the build parameters used to create the toolchain. Ex.: use uClibc instead of glibc.

8.2. SDK generation

- Building the SDK:

```
$ . build/envsetup.sh
```

```
$ lunch sdk-eng
```

```
$ make sdk
```

- If API modified, do this before make:

```
$ make update-api
```

- Location: [aosp]/out/host/linux-x86/sdk/

- Using a custom SDK:

- Eclipse->Window->Preferences->Android->"SDK Location"
- Eclipse->Window->"Android SDK and AVD Manager"->"Installed Packages"->"Update All..."

8.3. NDK generation

- Build

```
$ cd ndk/build/tools
```

```
$ export ANDROID_NDK_ROOT=[aosp]/ndk
```

```
$ ./make-release --help
```

```
$ ./make-release
```

IMPORTANT WARNING !!

This script is used to generate an NDK release package from scratch for the following host platforms: linux-x86

This process is EXTREMELY LONG and may take SEVERAL HOURS on a dual-core machine. If you plan to do that often, please read docs/DEVELOPMENT.TXT that provides instructions on how to do that more easily.

Are you sure you want to do that [y/N]

9. Compatibility Test Suite

- Android Compatibility Program:
 - Source code to Android stack
 - Compatibility Definition Document (CDD) – Policy
 - Compatibility Test Suite (CTS) – Mechanism
- Each Android version has own CDD & CTS
- CTS:
 - Part of AOSP
 - Run from host using USB over to attached device
 - Based on JUnit
 - Runs various test apps on target
 - Relies on ADB
 - Provides report to be analyzed and/or sent back to Google



On your machine

Download and
install the CTS.

Run the CTS.

Store and view
results.

Devices you attach
to your machine



Test
execution

Test
results

- Report:
 - .zip file containing XML files and screen-shots
 - Sent to: cts@android.com
- Building the CTS:

```
$ . build/envsetup.sh  
$ make cts
```
- Launching the CTS:

```
$ cd out/host/linux-x86/bin/  
$ ./cts  
$ cts_host >  
$ cts_host > help  
...
```

- Using the CTS:

```
$ cts_host > ls --plan
```

List of plans (8 in total):

Signature

RefApp

VM

Performance

AppSecurity

Android

Java

CTS

```
$ ./cts start --plan CTS
```

- Areas covered:
 - Signature tests
 - Platform API tests
 - Dalvik VM tests
 - Platform Data Model
 - Platform Intents
 - Platform Permissions
 - Platform Resources

10. Basic Hacks

- Add device
- Add app
- Add app overlay
- Add native tool or daemon
- Add native library

10.1. Adding a new device

- Create directory in device/:
 - device/acme/coyotepad/
- Makefile checklist:
 - AndroidProducts.mk
 - full_coyotepad.mk
 - BoardConfig.mk
 - Android.mk
- Menu integration:
 - vendorsetup.sh

10.1.1. AndroidProducts.mk

```
PRODUCT_MAKEFILES := \  
$(LOCAL_DIR)/full_coyotepad.mk
```

10.1.2. full_coyotepad.mk

```
$(call inherit-product, $(SRC_TARGET_DIR)/product/languages_full.mk)

# If you're using 4.2/Jelly Bean, use full_base.mk instead of
full.mk

$(call inherit-product, $(SRC_TARGET_DIR)/product/full.mk)

DEVICE_PACKAGE_OVERLAYS :=

PRODUCT_PACKAGES +=

PRODUCT_COPY_FILES +=

PRODUCT_NAME := full_coyotepad

PRODUCT_DEVICE := coyotepad

PRODUCT_MODEL := Full Android on CoyotePad, meep-meep
```

10.1.3. BoardConfig.mk

```
TARGET_NO_KERNEL := true  
TARGET_NO_BOOTLOADER := true  
TARGET_CPU_ABI := armeabi  
BOARD_USES_GENERIC_AUDIO := true  
USE_CAMERA_STUB := true
```

10.1.4. Android.mk

```
LOCAL_PATH := $(call my-dir)

include $(CLEAR_VARS)

ifneq ($(filter coyotepad,$(TARGET_DEVICE)),)
include $(call all-makefiles-under,$(LOCAL_PATH))
endif
```

10.1.5. vendorsetup.sh

```
add_lunch_combo full_coyotepad-eng
```

10.1.4. Android.mk

```
LOCAL_PATH := $(call my-dir)

include $(CLEAR_VARS)

ifneq ($(filter coyotepad,$(TARGET_DEVICE)),)
include $(call all-makefiles-under,$(LOCAL_PATH))
endif
```

10.2. Adding new applications

- Can use Eclipse to create initial version
- Copy Eclipse project to packages/apps
- Local:
 - Add app to device/acme/coyotepad/
 - Add Android.mk to app
 - Add to PRODUCT_PACKAGES in full_coyotepad.mk
- Global:
 - Add application in [aosp]/packages/apps
 - Add an appropriate Android.mk file to project
 - Add project to PRODUCT_PACKAGES in [aosp]/build/target/product/core.mk

```
LOCAL_PATH:= $(call my-dir)
include $(CLEAR_VARS)
LOCAL_MODULE_TAGS := optional
LOCAL_SRC_FILES := $(call all-java-files-under, src)
LOCAL_PACKAGE_NAME := HelloWorld
include $(BUILD_PACKAGE)
```


10.3. Adding an app overlay

- `device/acme/coyotepad/overlay/`
- `full_coyotepad.mk`:
 - `DEVICE_PACKAGE_OVERLAYS := device/acme/coyotepad/overlay`

10.4. Adding a native tool or daemon

- Local:
 device/acme/coyotepad/
- Global:
 - system/core/
 - system/
 - frameworks/base/cmds/
 - frameworks/native/cmds/
 - external/

```
LOCAL_PATH:= $(call my-dir)
include $(CLEAR_VARS)
LOCAL_MODULE := hello-world
LOCAL_MODULE_TAGS := optional
LOCAL_SRC_FILES := hello-world.cpp
LOCAL_SHARED_LIBRARIES := liblog
include $(BUILD_EXECUTABLE)
```

10.5. Add a native library

- Local:
device/acme/coyotepad/
- Global:
 - system/core/
 - frameworks/base/libs/
 - frameworks/native/libs/
 - external/
- In ≤ 2.3 :
 - See build/core/prelink-linux-arm.map

```
LOCAL_PATH:= $(call my-dir)
include $(CLEAR_VARS)
LOCAL_MODULE := libmylib
LOCAL_MODULE_TAGS := optional
LOCAL_PRELINK_MODULE := false
LOCAL_SRC_FILES := $(call all-c-files-under,.)
include $(BUILD_SHARED_LIBRARY)
```

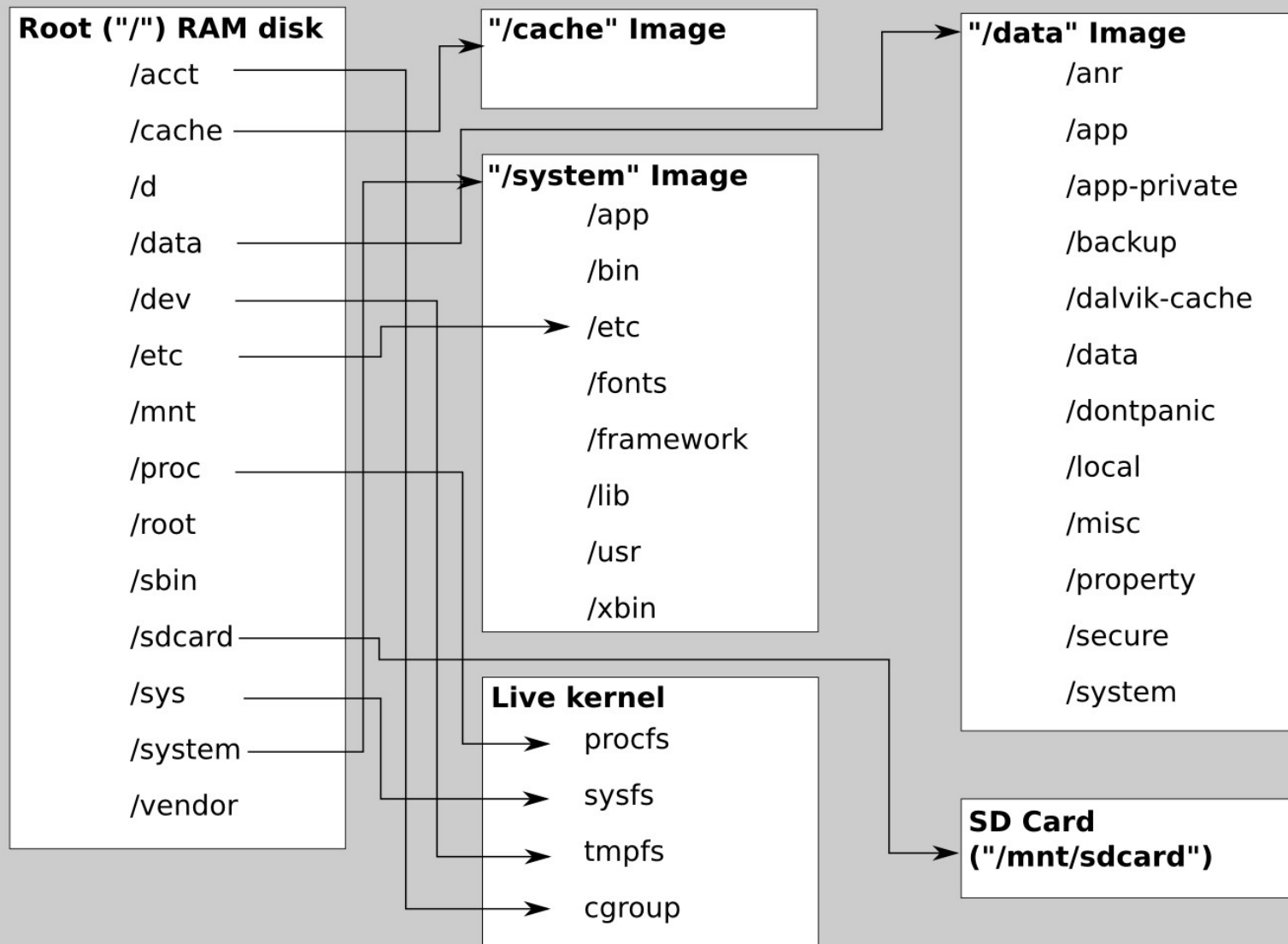
Kernel Selection

- Google:
 - <http://android.googlesource.com>
- Vanilla:
 - <http://www.kernel.org>
- ~~Either way ... you're screwed:~~
 - ~~Android kernel is a fork~~
 - ~~No resolution in sight~~
 - ~~**Cannot** use vanilla kernel as is ... androidisms~~
- Recent kernel have most Androidisms
- Android team continues to add stuff in their fork
- Talk to your SoC vendor

Native Android User-Space

- Filesystem layout
- Architecture vs. filesystem
- Build system and filesystem
- adb
- Command line
- Init
- ueventd

1. Filesystem layout



1.1. Root directory

- /acct => Control Group mount point (Documentation/cgroups.txt)
- /cache => cache flash partition
- /d => Symlink to /sys/kernel/debug
- /data => Android's “/data” filesystem
- /dev => Device nodes
- /etc => Symlink to /system/etc
- /mnt => Temporary mount point
- /proc => procfs
- /root => unused
- /sbin => eventd and adbd
- /sdcard => SD card mountpoint
- /sys => sysfs
- /system => Android's “/system” filesystem
- /vendor => Symlink to /system/vendor

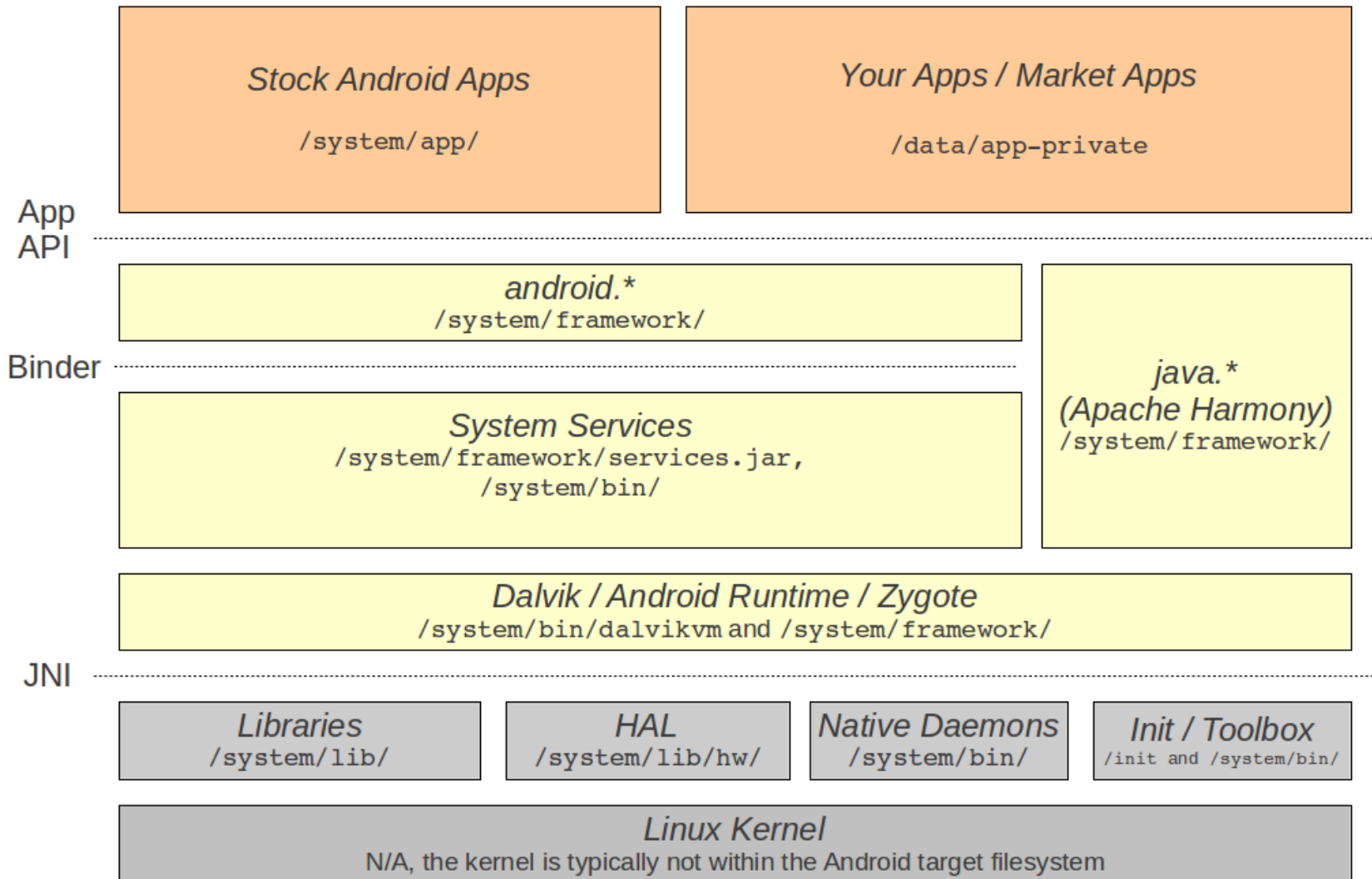
1.1. /system

- /app => Stock apps installed
- /bin => Native binaries and daemons
- /etc => Configuration files
- /fonts => TTFs
- /framework => Android framework .jar files
- /lib => Native libraries
- /usr => Miniature “/usr”
- /xbin => Optional/Extra binaries

1.2. /data

- /anr => ANR traces
- /app => App install location
- /app-private => Protected apps
- /backup => For Backup Manager
- /dalvik-cache => Dalvik DEX cache
- /data => App data
- /dontpanic => Last panic output (console + threads) for “dumpstate”
- /local => Shell-writable space
- /misc => Misc. data (wifi, vpn, bluetooth, ...)
- /property => Persistent system properties (country, lang., ...)
- /secure => Secure Android data available
- /system => System data

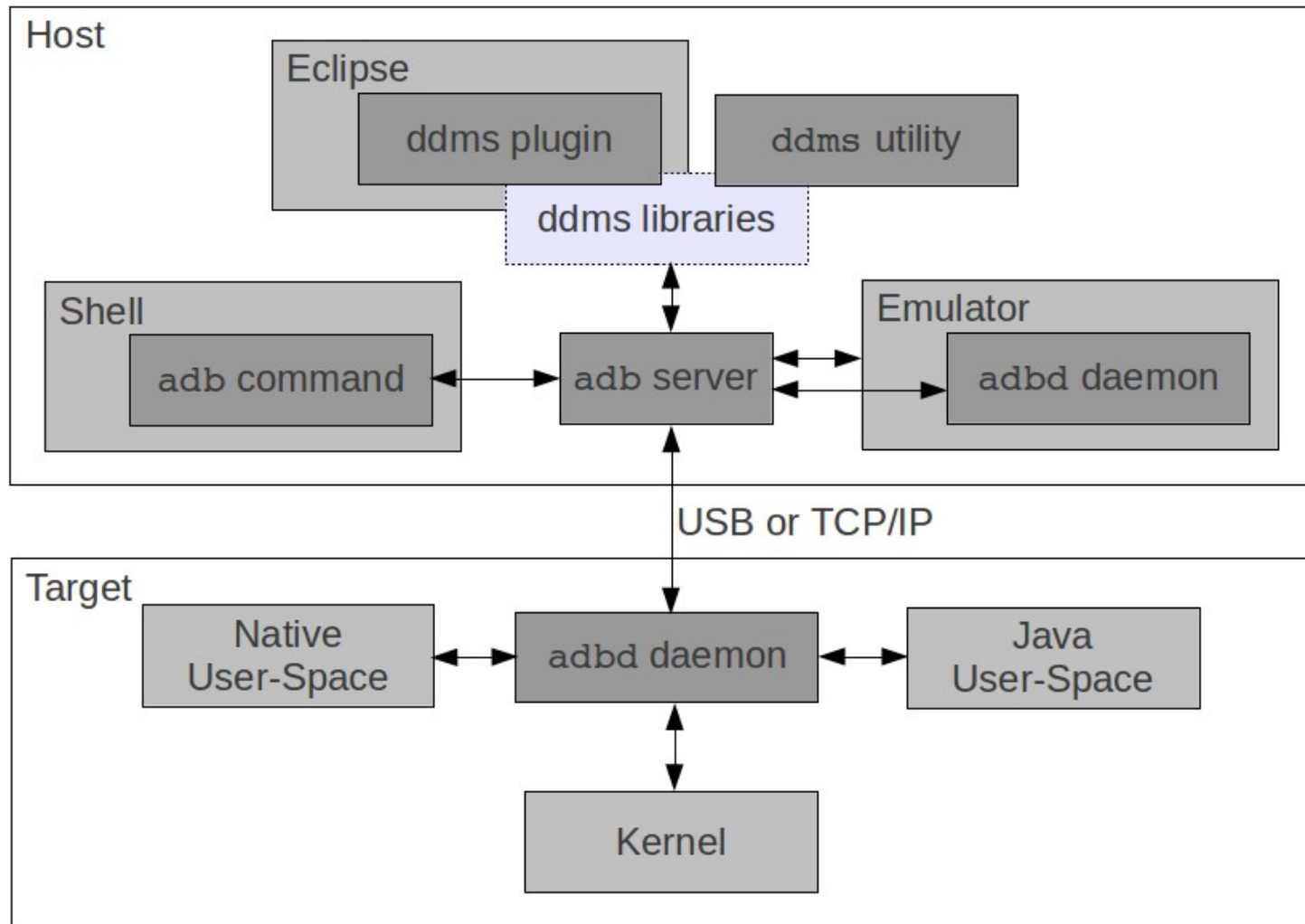
2. Architecture vs. filesystem



3. Build system and filesystem

- Build templates
 - BUILD_EXECUTABLE = /system/bin
 - BUILD_SHARED_LIBRARY = /system/lib
 - BUILD_PACKAGE = /system/app
- Default rights and ownership
 - system/core/include/private/android_filesystem_config.h
 - User IDs / names
 - android_dirs struct
 - android_files struct

4. adb



4.1. What adb can do

- See online help for full details
- Managing device connections
- Remote commands:
 - shell
 - log dumping
 - bug reports
 - port forwarding
 - dalvik debugging

- Filesystem commands
 - push
 - pull
 - sync
 - install / uninstall
- State-altering commands
 - reboot
 - run as root
 - switching connection type
 - controlling the emulator
- Tunneling PPP

5. Command line

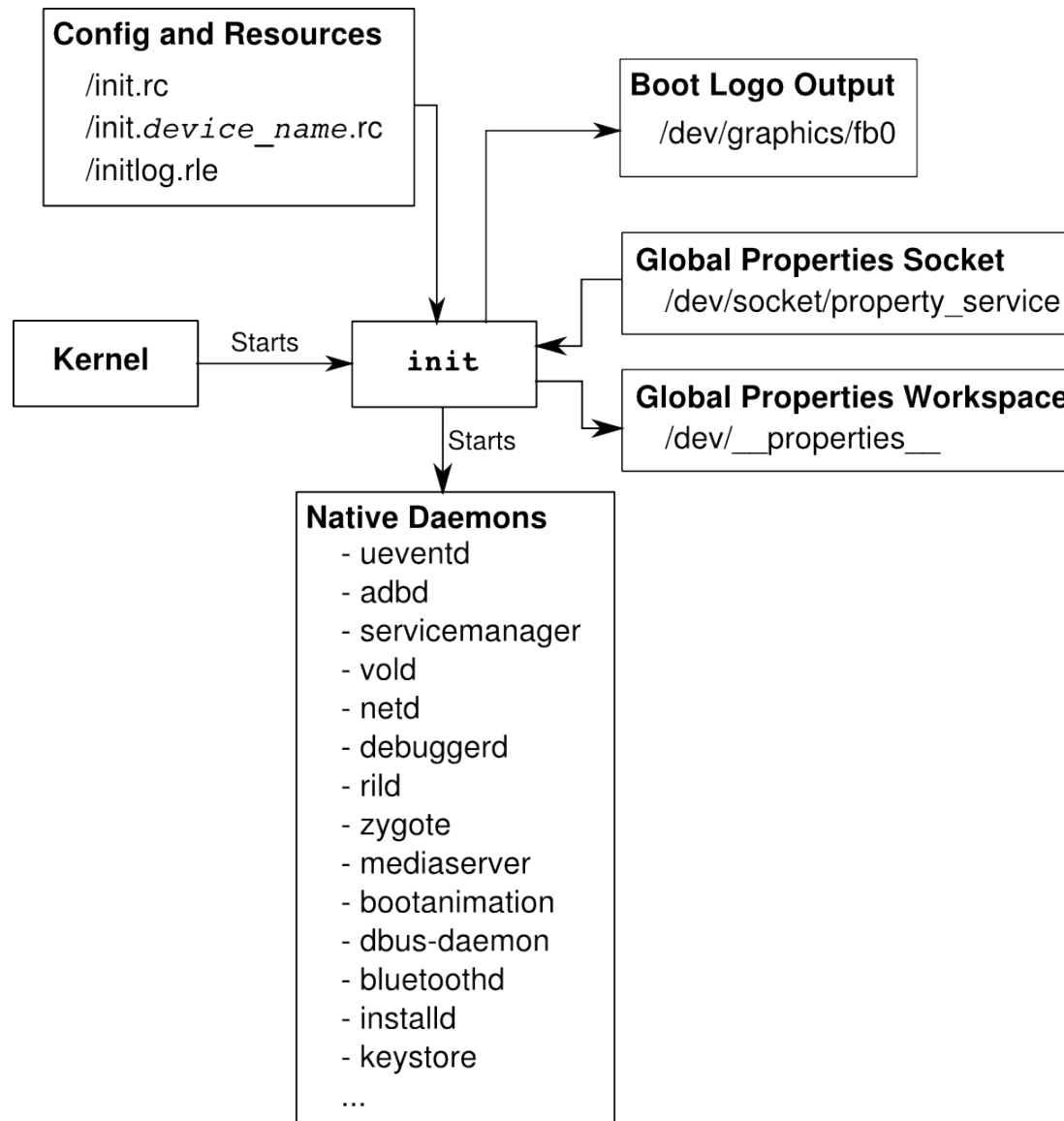
- Shell:
 - NetBSD shell up to 2.3/GB
 - MirBSD Korn shell since 4.0/ICS
- Toolbox:
 - Traditional Linux commands
 - Global properties
 - Input events
 - Services control
 - Logging
 - ioctl
 - Device wiping
 - etc.

- In aosp:
 - /system/core/toolbox
- In filesystem:
 - /system/bin/toolbox
- Provides

alarm date getevent insmod ls mv powerd renice schedtop
 smd top dd getprop ioctl lsmod nandread printenv rm
 sendevent start umount cat hd ionice lsof netstat ps rmdir
 setconsole stop uptime chmod df id kill mkdir newfs_msdos r
 mmmod setkey sync vmstat chown dmesg ifconfig ln readtty
 rotatefb setprop syren watchprops cmp exists iftop log mount
 notify reboot route sleep wipe

- logcat
- netcfg
- debuggerd

6. Init



6.1. Config files

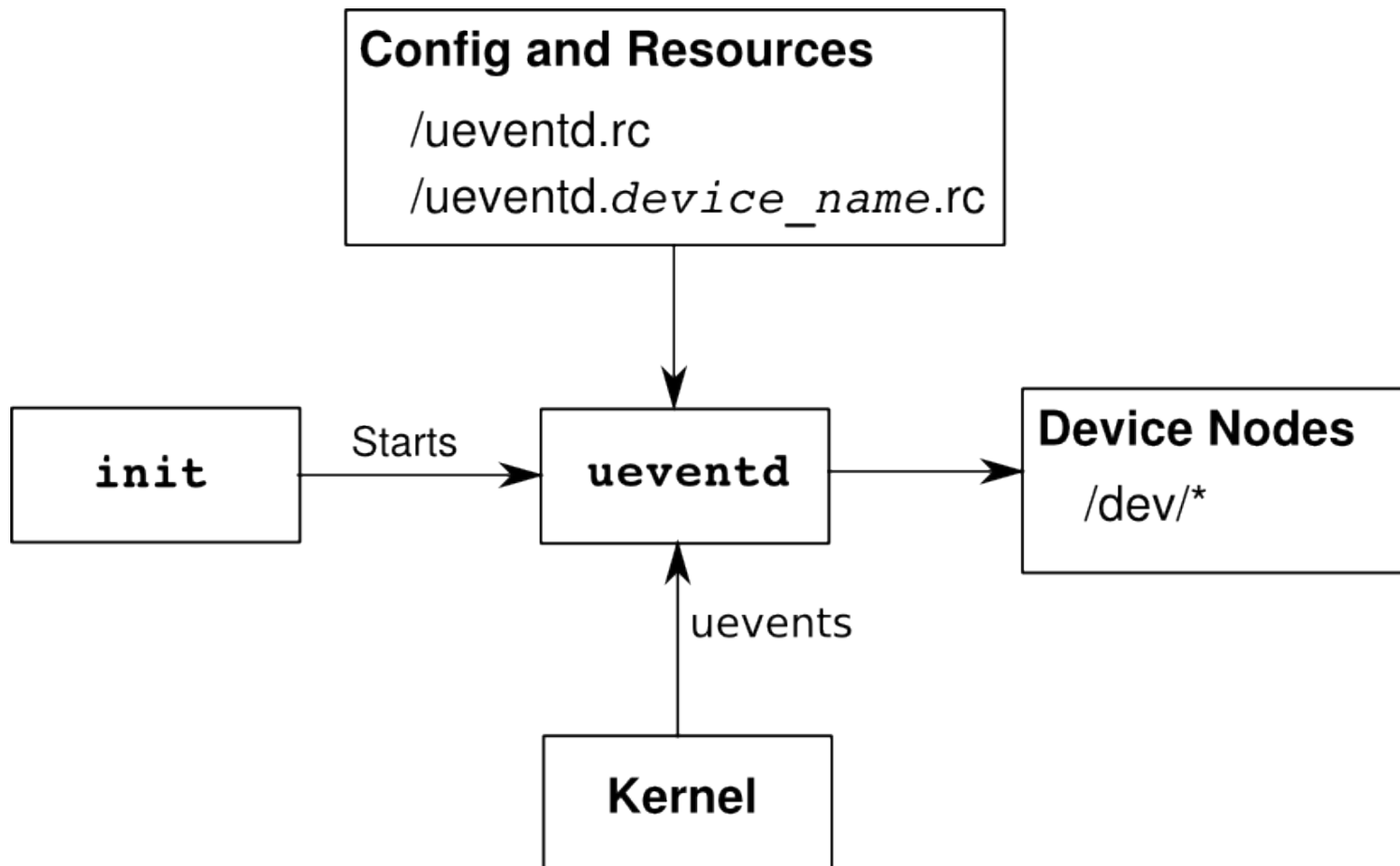
- Location:
 - /init.rc
 - /init.[board].rc
- Semantics:
 - actions
 - Property triggers
 - Commands
 - services (not related to system services or apps)
 - Parameters

6.2. Global properties

- /dev/socket/property_service
- /dev/__properties__
- foo.bar.property
- ro.*
- persist.* => /data/property
- ctl.*
- net.change

- Std files:
 - /system/build.prop
 - /default.prop
- Other files:
 - /system/default.prop
 - /data/local.prop
- Code:
 - `property_get("dalvik.vm.heapsize", heapsizeOptsBuf+4, "16m");`

7. ueventd



7.1. /ueventd.rc

/dev/null 0666 root root

/dev/zero 0666 root root

/dev/full 0666 root root

/dev/ptmx 0666 root root

/dev/tty 0666 root root

System Services and Framework Internals

1. Kickstarting the Framework
2. Utilities and Commands
3. Native Daemons
4. System Services Internals
5. Hardware Abstraction Layer
6. Android Framework

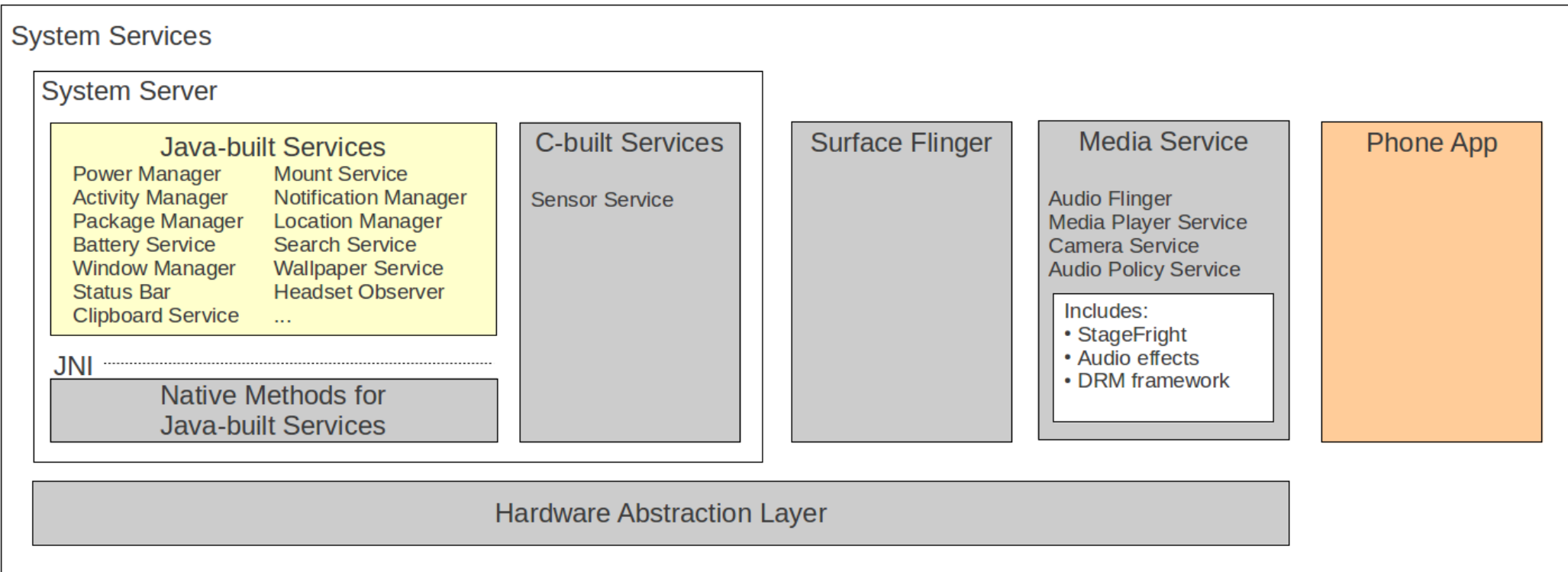
1. Kickstarting the Framework

- Core building blocks
- System services
- Boot animation
- Dex optimization
- Apps startup

1.1. Core building blocks

- Service manager
- Dalvik
- Android runtime
- Zygote

1.2. System services



1.3. Boot animation

- Default boot animation
- bootanimation.zip
- setprop debug.sf.nobootanimation 1

1.4. Dex optimization

- Files:
 - .jar
 - .apk
- Directories:
 - BOOTCLASSPATH
 - /system/etc/permission/platform.xml
 - /system/framework
 - /system/app
 - /vendor/app
 - /data/app
 - /data/app-private

1.5. Apps startup

- Input methods
- Persistent apps
- Home screen
- BOOT_COMPLETED intent
- APPWIDGET_UPDATE intent

2. Utilities and Commands

- General-purpose commands
 - service
 - dumphsys
 - dumpstate
 - rawbu
- Service-specific utilities
 - am
 - pm
 - svc

- ime
- input
- monkey
- bmgr
- stagefright
- Dalvik utilities
 - dalvikvm
 - dvz
 - dexdump

3. Native daemons

- servicemanager
- installd
- vold
- netd
- rild
- keystore

3.1. servicemanager

- In aosp:
 - /frameworks/base/cmds/servicemanager/
- In filesystem:
 - /system/bin/
- Provides:
 - Context management for binder
 - Service index for entire system

3.2. installd

- In aosp:
 - /frameworks/base/cmds/installd
- In filesystem:
 - /system/bin/
- Provides:
 - Package install/uninstall
 - Sanity checks and verifications
 - Interfaces with Package Manager service

3.3. vold

- In aosp:
 - /system/vold/
- In filesystem:
 - /system/bin/
- Provides:
 - Volume mounter
 - Auto-mount
 - Auto-format mounted devices

3.4. netd

- In aosp:
 - /system/netd/
- In filesystem:
 - /system/bin/
- Provides:
 - Management of aspects of networking
 - Interfaces with Network Management service

3.5. rild

- In aosp:
 - /hardware/ril/mock-ril/
- In filesystem:
 - /system/bin/
- Provides:
 - “Radio Interface Layer” to phone hardware

3.6. keystore

- In aosp:
 - frameworks/base/cmds/keystore/
- In filesystem:
 - /system/bin
- Provides:
 - Store key-value pairs
 - Security keys for connecting to network infrastructure

4. System Services Internals

- Services run by System Server
- Observing the System Server
- Calling on system services
- Inside a few system services
- Creating your own system service

4.1. Services run by the System Server

Entropy Service
Power Manager
Activity Manager
Telephone Registry
Package Manager
Account Manager
Content Manager
System Content Providers
Battery Service
Lights Service
Vibrator Service
Alarm Manager
Init Watchdog
Sensor Service
Window Manager
Bluetooth Service

Device Policy
Status Bar
Clipboard Service
Input Method Service
NetStat Service
NetworkManagement Service
Connectivity Service
Throttle Service
Accessibility Manager
Mount Service
Notification Manager
Device Storage Monitor
Location Manager
Search Service
DropBox Service
Wallpaper Service

Audio Service
Headset Observer
Dock Observer
UI Mode Manager Service
Backup Service
AppWidget Service
Recognition Service
Status Bar Icons
DiskStats Service
ADB Settings Observer

4.1.1. Some stats

- frameworks/base/services/java/com/android/server:
 - 8.1 M
 - ~300 files
 - 190 kloc
- Activity manager:
 - 1.6M
 - 45+ files
 - 35+ kloc

4.2. Observing the System Server

- Logcat
- dumpsys

4.2.1. logcat

- Find the System Server's PID

```
$ adb shell ps | grep system_server
```

```
system    63    32    120160 35408 ffffffff afd0c738 S system_server
```

- Look for its output:

```
$ adb logcat | grep "63)"
```

```
...
D/PowerManagerService( 63): bootCompleted
I/TelephonyRegistry( 63): notifyServiceState: 0 home Android Android 310260 UMTS CSS not supp...
I/TelephonyRegistry( 63): notifyDataConnection: state=0 isDataConnectivityPossible=false reason=null
interfaceName=null networkType=3
I/SearchManagerService( 63): Building list of searchable activities
I/WifiService( 63): WifiService trying to setNumAllowed to 11 with persist set to true
I/ActivityManager( 63): Config changed: { scale=1.0 imsi=310/260 loc=en_US touch=3 keys=2/1/2 nav=3/1 ...
I/TelephonyRegistry( 63): notifyMessageWaitingChanged: false
I/TelephonyRegistry( 63): notifyCallForwardingChanged: false
I/TelephonyRegistry( 63): notifyDataConnection: state=1 isDataConnectivityPossible=true reason=simL...
I/TelephonyRegistry( 63): notifyDataConnection: state=2 isDataConnectivityPossible=true reason=simL...
D/Tethering( 63): MasterInitialState.processMessage what=3
I/ActivityManager( 63): Start proc android.process.media for broadcast
com.android.providers.downloads/.DownloadReceiver: pid=223 uid=10002 gids={1015, 2001, 3003}
I/RecoverySystem( 63): No recovery log file
W/WindowManager( 63): App freeze timeout expired.
...
```

4.2.2. dumphsys

Currently running services:

SurfaceFlinger

accessibility

account

activity

alarm

appwidget

audio

backup

...

wifi

window

DUMP OF SERVICE SurfaceFlinger:

+ Layer 0x396b90

z= 21000, pos=(0, 0), size=(480, 800), needsBlending=1, needsDithering=1, invalidat ...

0]

name=com.android.launcher/com.android.launcher2.Launcher

client=0x391e48, identity=6

[head= 1, available= 2, queued= 0] reallocMask=00000000, inUse=-1, identity=6, status=0

format= 1, [480x800:480] [480x800:480], freezeLock=0x0, dq-q-time=53756 us

...

4.3. Calling on System Services

- Use getSystemService
- Ex: NotificationManager Object reference:

```
String ns = Context.NOTIFICATION_SERVICE;
```

```
NotificationManager mNotificationManager = (NotificationManager) \  
getSystemService(ns);
```

- Prepare your content
- Call on the object:

```
mNotificationManager.notify(HELLO_ID, notification);
```


4.4. Inside a few System Services

- Get the AOSP ... repo, etc.
- Tricks:
 - Import into Eclipse and collapse methods
 - Use reverse-engineering tools:
 - Imagix
 - Rationale
 - Lattix
 - Scitools
 - ...
- Be patient, this isn't documented anywhere ...

4.4.1. ActivityManager

- Start new Activities, Services
- Fetch Content Providers
- Intent broadcasting
- OOM adj. maintenance
- Application Not Responding
- Permissions
- Task management
- Lifecycle management

- Ex. starting new app from Launcher:
 - onClick(Launcher)
 - startActivity(Activity.java)
 - *<Binder>*
 - ActivityManagerService
 - startViaZygote(Process.java)
 - *<Socket>*
 - Zygote

4.4.2. Package Manager

- 19 kloc
- 830 K
- Installation / removal
- Permissions
- Intent resolution (also IntentResolver.java)
- Called by Activity Manager

4.4.3. Window Manager

- Main thread
- Window manipulation
- Wallpaper handling
- Orientation
- Focus
- Layering
- Input event management

4.4.4. Notification Manager

- Toasts
- Notifications
- Sound playback (see NotificationPlayer.java)

4.4.5. Power Manager

- Wakelocks
- Sleep
- Brightness
- Lock

4.4.6. Network Management Service

- Talks to “netd” /system/netd
- Interface configuration
- Tethering
- DNS

4.4.7. Mount Service

- Mount / Unmount
- Format
- USB mass storage
- OBB

4.4.8. Location Manager

- Manage location providers
- `getBestProvider()`
- Proximity alerts
- Last known location

4.4.9. Status Bar Manager

- Expand / collapse
- Icon visibility
- Reveal callbacks
- Callbacks for notification manager

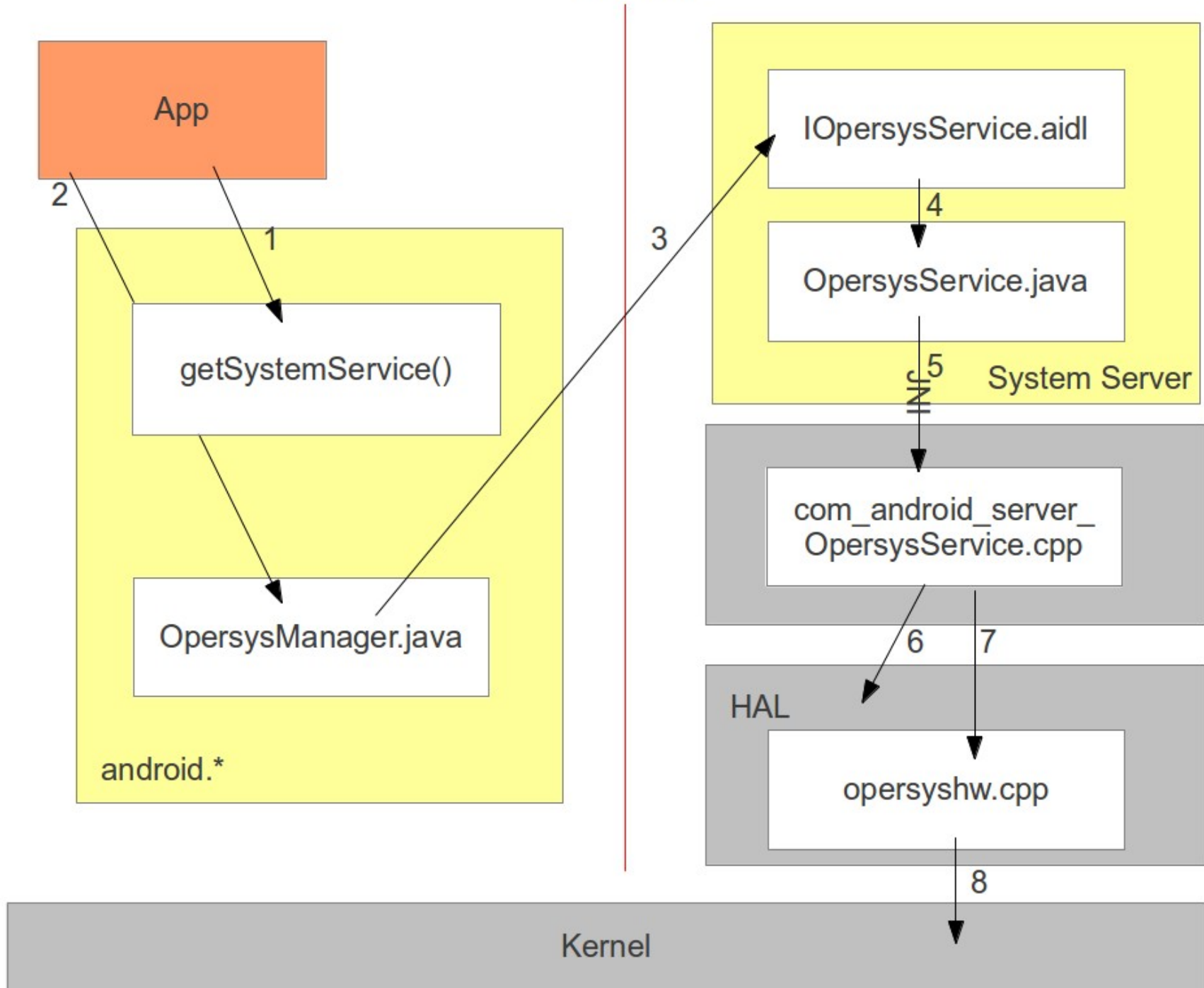
4.4.10. Backup Manager

- Enable / disable
- Transport management
- backupNow()
- ...

4.5. Creating your own System Service

- Add your code to:
frameworks/base/services/java/com/android/server/
- Have the SystemServer.java init+reg. your service
- Define hardware API for apps
- Expose through:
 - frameworks/base/core/java/android/os/[server].aidl
- Call on native “driver” code through JNI
- Implement or connect to appropriate driver
- Create an app that calls on service
- May need to create new SDK ...

BINDER



4.5.1. OpersysService.java

```
package com.android.server;

import android.content.Context;
import android.os.Handler;
import android.os.IOpersysService;
import android.os.Looper;
import android.os.Message;
import android.os.Process;
import android.util.Log;

public class OpersysService extends IOpersysService.Stub {
    private static final String TAG = "OpersysService";
    private OpersysWorkerThread mWorker;
    private OpersysWorkerHandler mHandler;
    private Context mContext;

    public OpersysService(Context context) {
        super();
        mContext = context;
        mWorker = new OpersysWorkerThread("OpersysServiceWorker");
        mWorker.start();
        Log.i(TAG, "Spawned worker thread");
    }

    public void setValue(int val) {
        Log.i(TAG, "setValue " + val);
        Message msg = Message.obtain();
        msg.what = OpersysWorkerHandler.MESSAGE_SET;
        msg.arg1 = val;
        mHandler.sendMessage(msg);
    }
}
```

```

private class OpersysWorkerThread extends Thread{
public OpersysWorkerThread(String name) {
    super(name);
}

public void run() {
    Looper.prepare();
    mHandler = new OpersysWorkerHandler();
    Looper.loop();
}

private class OpersysWorkerHandler extends Handler {
private static final int MESSAGE_SET = 0;

@Override
public void handleMessage(Message msg) {
    try {
        if (msg.what == MESSAGE_SET) {
            Log.i(TAG, "set message received: " + msg.arg1);
        }
    }
    catch (Exception e) {
        // Log, don't crash!
        Log.e(TAG, "Exception in OpersysWorkerHandler.handleMessage:", e);
    }
}
}
}

```


4.5.2. IOpersysService.aidl

```
package android.os;
/**
 * {@hide}
 */
interface IOpersysService {
void setValue(int val);
}
```

4.5.3. frameworks/base/Android.mk

...

```
core/java/android/os/IPowerManager.aidl \  
core/java/android/os/IOperSysService.aidl \  
core/java/android/os/IRemoteCallback.aidl \  

```

...

4.5.4. SystemServer.java

Should eventually be Context.OPERSYS_SERVICE

...

```
try {  
    Slog.i(TAG, "Opersys Service");  
    ServiceManager.addService("opersys", new OpersysService(context));  
} catch (Throwable e) {  
    Slog.e(TAG, "Failure starting OpersysService Service", e);  
}
```

...

4.5.5. HelloServer.java

```
package com.opersys.helloserver;

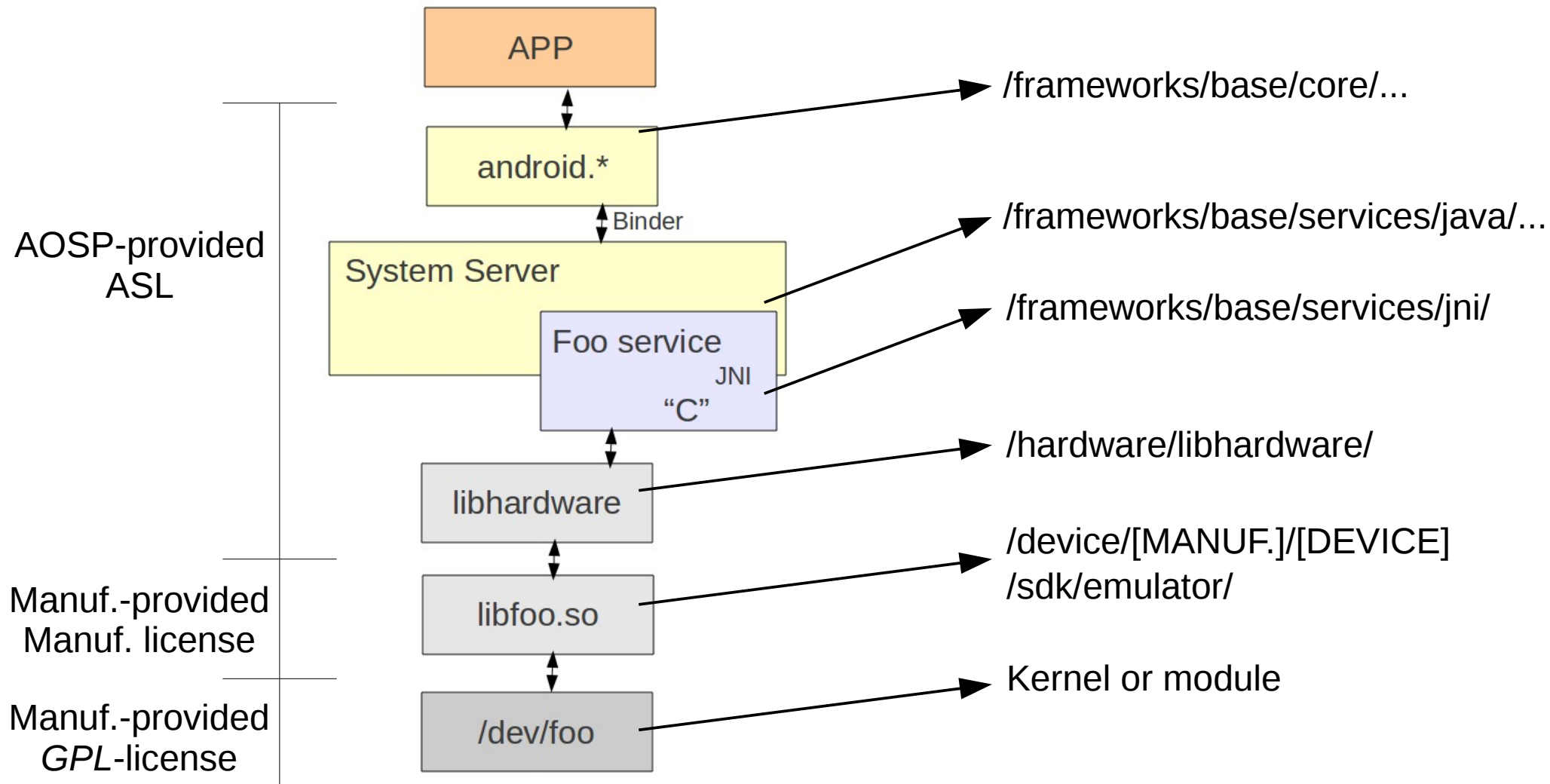
import android.app.Activity;
import android.os.Bundle;
import android.os.ServiceManager;
import android.os.IOpersysService;
import android.util.Log;

public class HelloServer extends Activity {
    private static final String DTAG = "HelloServer";

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        IOpersysService om =
        IOpersysService.Stub.asInterface(ServiceManager.getService("opersys"));
        try {
            Log.d(DTAG, "Going to call service");
            om.setValue(20);
            Log.d(DTAG, "Service called succesfully");
        }
        catch (Exception e) {
            Log.d(DTAG, "FAILED to call service");
            e.printStackTrace();
        }
    }
}
```

5. Hardware Abstraction Layer



- [aosp]/hardware/libhardware/include/hardware
 - gps.h
 - lights.h
 - sensors.h
- [aosp]/hardware/ril/include/telephony/
 - ril.h
- Examples in [aosp]/device/samsung/crespo/
 - libaudio
 - libcamera
 - liblight
 - libsensors
- Using JNI to call C functions

6. Android Framework

- Location and components
- android.*
- Customization

6.1. Location and components

- [aosp]/frameworks/base
 - /cmds => native cmds and daemons
 - /core => android.* and com.android.*
 - /data => Fonts and sounds
 - /graphics => 2D & Renderscript
 - /include => "C" includes
 - /keystore => security key store
 - /libs => "C" libraries
 - /location => Location provider
 - /media => Stagefright, codecs, etc.
 - /native => Native code for some frameworks components
 - /obex => Bluetooth obex
 - /opengl => GL library and java code
 - /packages => A few core packages (Status Bar)
 - /services => System server
 - /telephony => Phone related functionality
 - /tools => A few core tools (aapt, aidl, ...)
 - /voip => RTP & SIP interfaces
 - /vpn => VPN functionality
 - /wifi => Wifi manager, monitor, etc.

6.2. android.*

accessibilityservice	content	hardware	pim	speech
accounts	database	inputmethodservice	preference	test
annotation	ddm	net	provider	text
app	debug	nfc	security	util
appwidget	emoji	os	server	view
bluetooth	gesture	service	webkit	widget

6.3. Customization

- Extending API
- Boot screen
- Status bar
- Network
- Preloaded apps
- Browser bookmarks
- Email provider customization
- Themes

6.3.1. Extending API – System service

- frameworks/base/core/java/android/
 - app/ContextImpl.java
 - content/Context.java
 - os/OpersysManager.java

6.3.1.1. app/ContextImpl.java

```
...  
import android.os.IOpsysService;  
import android.os.OpsysManager;  
...  
    registerService(OPERSYS_SERVICE, new ServiceFetcher() {  
        public Object createService(ContextImpl ctx) {  
            IBinder b =  
                ServiceManager.getService(OPERSYS_SERVICE);  
            IOpsysService service =  
                IOpsysService.Stub.asInterface(b);  
            return new OpsysManager(service);  
        }  
    });  
...
```

6.3.1.2. content/Context.java

```
...  
/**  
 * Use with {@link #getSystemService} to retrieve a  
 * {@link android.nfc.NfcManager} for using NFC.  
 *  
 * @see #getSystemService  
 */  
public static final String NFC_SERVICE = "nfc";  
  
/** The Opersys service */  
public static final String OPERSYS_SERVICE = "opersys";  
...
```

6.3.1.3. os/OpersysManager.java

```
package android.os;

import android.os.IOpersysService

public class OpersysManager
{
    public void setValue(int value)
    {
        try {
            mService.setValue(value);
        } catch (RemoteException e) {
        }
    }

    public OpersysManager(IOpersysService service)
    {
        mService = service;
    }

    IOpersysService mService;
}
```

6.3.2. Status bar

- Location:
 - frameworks/base/packages/SystemUI/src/com/android/systemui/statusbar
- Disable phone signal - SignalClusterView.java:

```
if (mMobileVisible && !mIsAirplaneMode) {  
    /* DISABLE  
    mMobileGroup.setVisibility(View.VISIBLE);  
    mMobile.setImageResource(mMobileStrengthId);  
    mMobileActivity.setImageResource(mMobileActivityId);  
    mMobileType.setImageResource(mMobileTypeId);  
    mMobileGroup.setContentDescription(mMobileTypeDescription + ...  
    */
```

- Disable battery meter -
policy/BatteryController.java:

```
/* DISABLE
for (int i=0; i<N; i++) {
    ImageView v = mIconViews.get(i);
    v.setImageResource(icon);
    v.setImageLevel(level);
    v.setContentDescription(mContext.getString(R.string.acces...)
}
*/
```


6.3.3. Network

- Locations:
 - Global static:
 - frameworks/base/core/res/res/xml/apns.xml
 - Device static:
 - `PRODUCT_COPY_FILES := vendor/acme/etc/apns-conf-us.xml:system/etc/apns-conf.xml`
 - Dynamic:
 - system/etc/apns-conf.xml
- Format:

```
<apn carrier="T-Mobile US"  
    mcc="310"  
    mnc="260"  
    apn="wap.voicestream.com"  
    user="none"  
    server="*"  
    password="none"  
    proxy="216.155.165.50"  
    port="8080"  
    mmsc="http://216.155.174.84/servlets/mms"  
>
```

6.3.4. Browser bookmarks

- See packages/apps/Browser/res/values/strings.xml

```
<!-- Bookmarks -->
```

```
<string-array name="bookmarks">
```

```
  <item>Google</item>
```

```
  <item>http://www.google.com/</item>
```

```
  <item>Yahoo!</item>
```

```
  <item>http://www.yahoo.com/</item>
```

```
  <item>MSN</item>
```

```
  <item>http://www.msn.com/</item>
```

```
  <item>MySpace</item>
```

```
  <item>http://www.myspace.com/</item>
```

```
...
```

6.3.5. Email provider customization

- See packages/apps/Email/res/xml/providers.xml

<!-- Gmail variants -->

```
<provider id="gmail" label="Gmail" domain="gmail.com">
  <incoming uri="imap+ssl+://imap.gmail.com" username="$email"/>
  <outgoing uri="smtp+ssl+://smtp.gmail.com" username="$email"/>
</provider>
<provider id="googlemail" label="Google Mail" domain="googlemail.com">
  <incoming uri="imap+ssl+://imap.googlemail.com" username="$email"/>
  <outgoing uri="smtp+ssl+://smtp.googlemail.com" username="$email"/>
</provider>
```

...

<!-- Common US providers -->

```
<provider id="aim" label="AIM" domain="aim.com">
  <incoming uri="imap://imap.aim.com" label="IMAP" username="$email"/>
  <outgoing uri="smtp://smtp.aim.com:587" username="$email"/>
</provider>
<provider id="aol" label="AOL" domain="aol.com">
  <incoming uri="imap://imap.aol.com" label="IMAP" username="$email"/>
  <outgoing uri="smtp://smtp.aol.com:587" username="$email"/>
</provider>
```

...

6.3.6. Themes

- See
`framework/base/core/res/res/values/styles.xml`

Thank you ...

karim.yaghmour@opersys.com



Acknowledgements:

- Some figures and snippets taken from Google's Android “Dev Guide” at developer.android.com distributed under the Apache 2.0 license.